Lab 3 (DNS), Week 2

Checkpoint

- Show me that you can parse a name with a pointer OR
- Show me that you can parse the IP address of the next server

- How to submit checkpoint:
  - Email me your output to kwebb1@swarthmore.edu with a subject that starts with [Lab3]
  - Meet with me on Monday, March 14 – walk-ins welcome:
    - 10:30 AM – noon
    - 2:30 PM – 5:00 PM

Where we are…

- I'm assuming that you:
  - Constructed a request
  - Sent the request
  - Received a valid reply
  - Are looking at the reply and don't know how to make sense of it
- If those aren't true, use Wireshark to dissect what you're sending
- Contact me if you can't get a valid reply

Timeouts

- See: *timeout.py*

```
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.settimeout(5)

# Send request to server

try:
    response, address = sock.recvfrom(4096)
    # Response is valid, use it

except socket.timeout as e:
    # recvfrom took five seconds, give up and try the next server
```

-m flag, MX Records

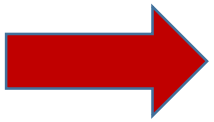
- You may assume that, if the -m flag appears, it will be in between the *lab3* and *host name*, e.g.,:
  - ./lab3 -m cs.swarthmore.edu

- If the -m flag is set, you should query for record type MX (15) rather than A (1).

- The answer you ultimately get will be a host name. Once you have that host name, resolve it by making queries for its corresponding A record.

Parsing Responses

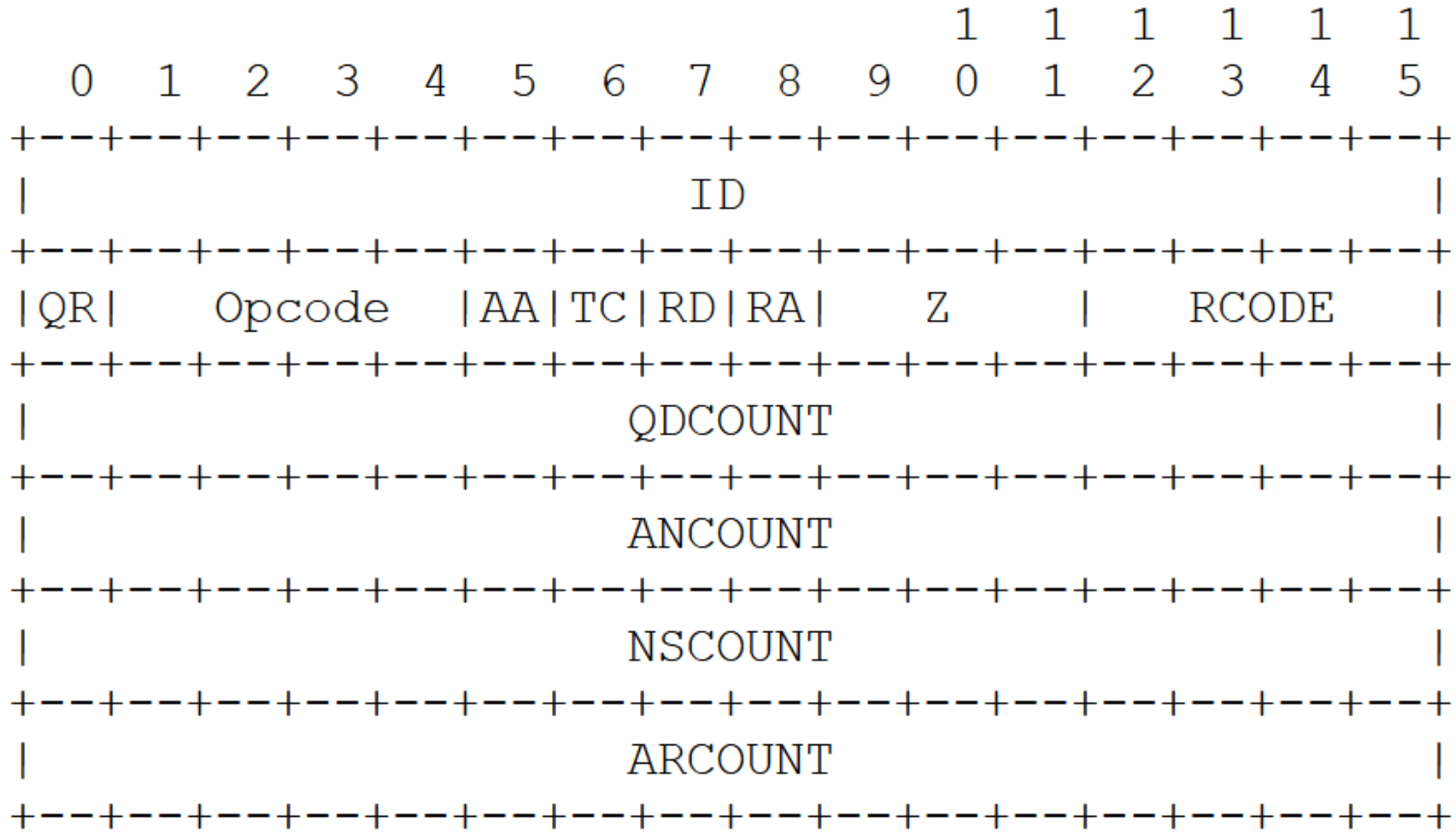
- So, you have a bunch of bytes…



```
+---------------------+
|        Header       |
+---------------------+
|       Question      | the question for the name server
+---------------------+
|        Answer       | RRs answering the question
+---------------------+
|      Authority      | RRs pointing toward an authority
+---------------------+
|      Additional     | RRs holding additional information
+---------------------+
```

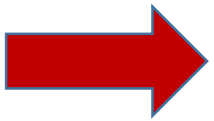
Parsing Header

- Same as the header you built, in reverse
- If you want to detect errors early, look at RCODE. Also ok to just look at the type of answer you get.



Parsing Responses

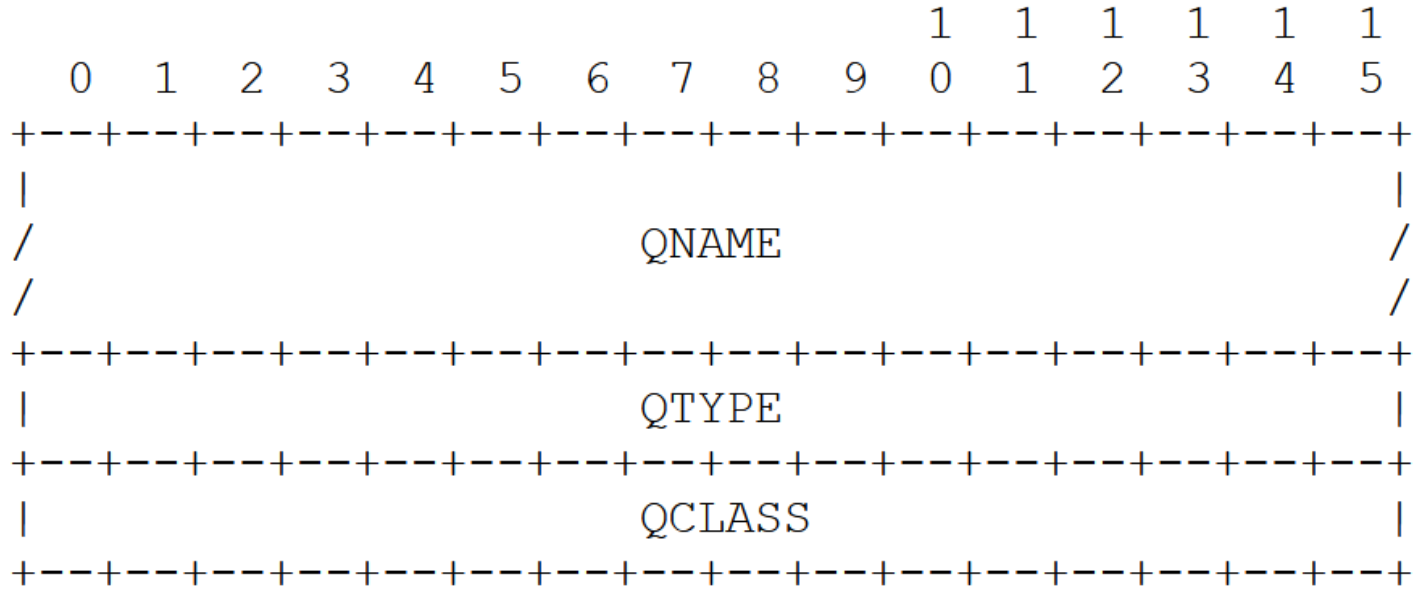
- So, you have a bunch of bytes…



```
+---------------------+
|        Header       |
+---------------------+
|       Question      | the question for the name server
+---------------------+
|        Answer       | RRs answering the question
+---------------------+
|      Authority      | RRs pointing toward an authority
+---------------------+
|      Additional     | RRs holding additional information
+---------------------+
```

Parsing Questions

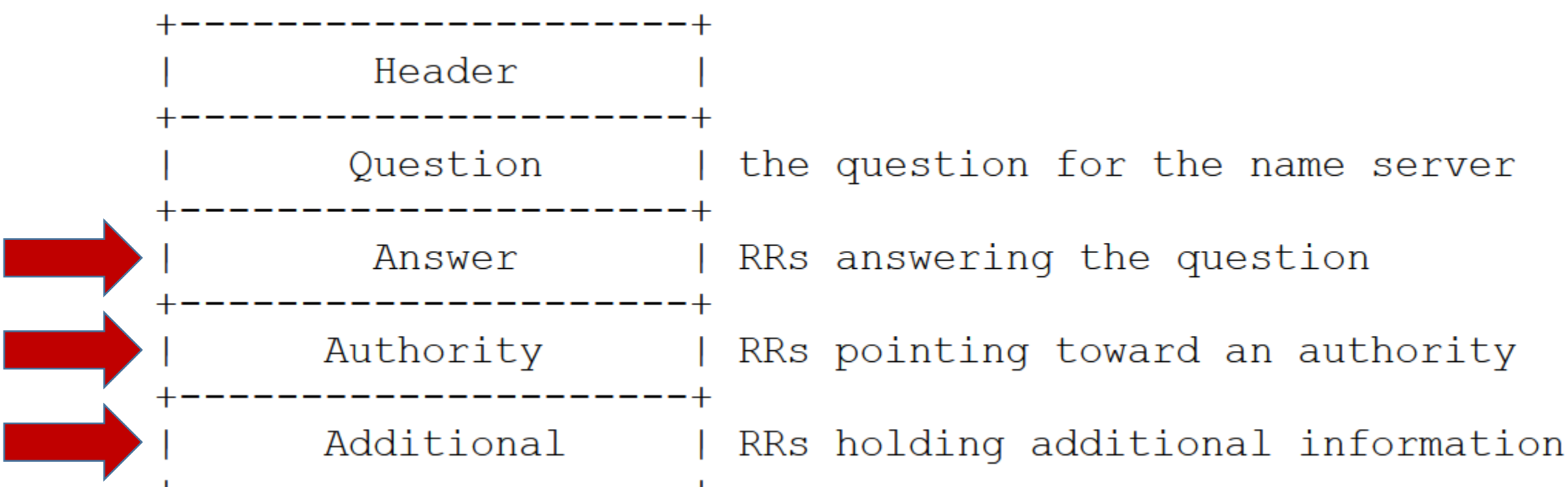
- Note the // on the border of QNAME – variable length
- You don't really care about what the question contains



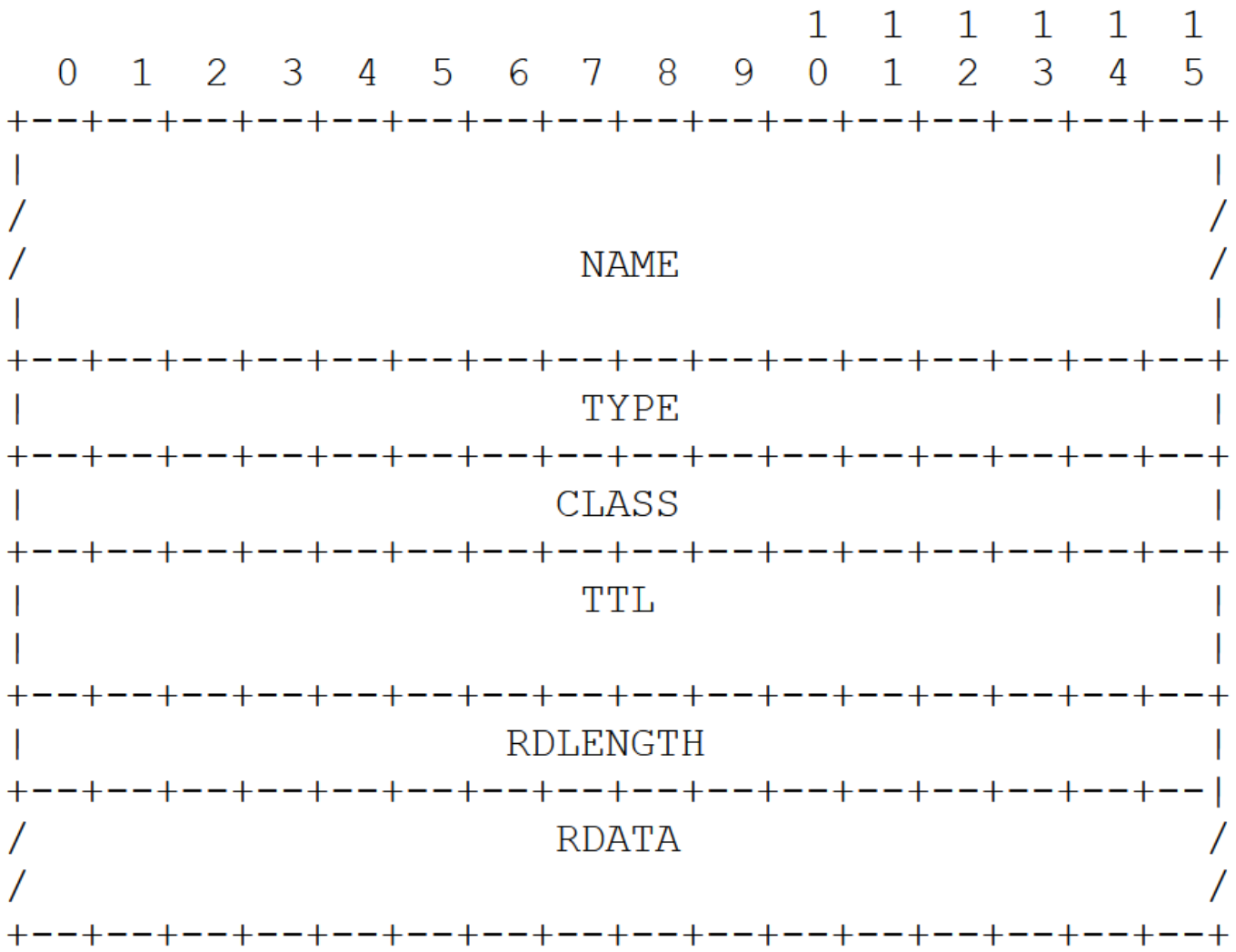
- You MUST know how many bytes are in the question to move beyond it

Parsing Responses

- So, you have a bunch of bytes…



Parsing Resource Records

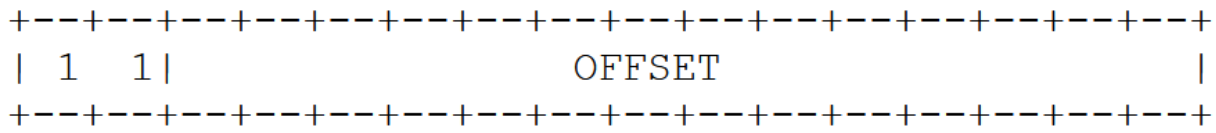


Names

4.1.4. Message compression

In order to reduce the size of messages, the domain system utilizes a compression scheme which eliminates the repetition of domain names in a message. In this scheme, an entire domain name or a list of labels at the end of a domain name is replaced with a pointer to a prior occurance of the same name.

The pointer takes the form of a two octet sequence:



The first two bits are ones. This allows a pointer to be distinguished from a label, since the label must begin with two zero bits because labels are restricted to 63 octets or less. (The 10 and 01 combinations are reserved for future use.) The OFFSET field specifies an offset from the start of the message (i.e., the first octet of the ID field in the domain header). A zero offset specifies the first byte of the ID field, etc.

The compression scheme allows a domain name in a message to be represented as either:

- a sequence of labels ending in a zero octet

- a pointer

- a sequence of labels ending with a pointer

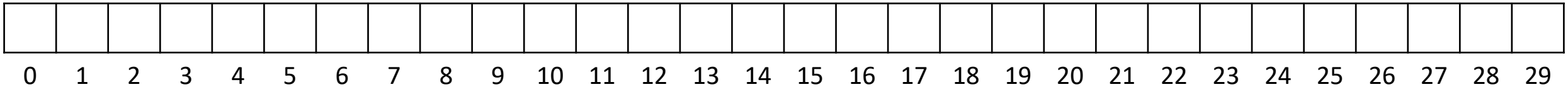
- The following is a real example of the response that I got from running:

```
dig @dns.cs.swarthmore.edu demo.cs.swarthmore.edu
```

Parsing Names (no pointers)

Binary

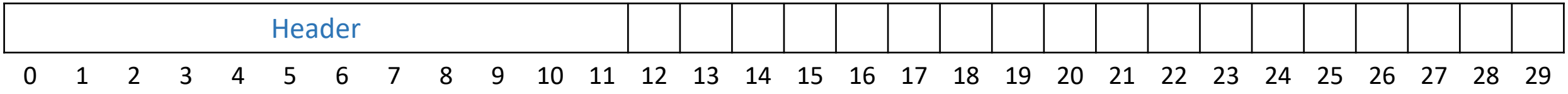
ASCII



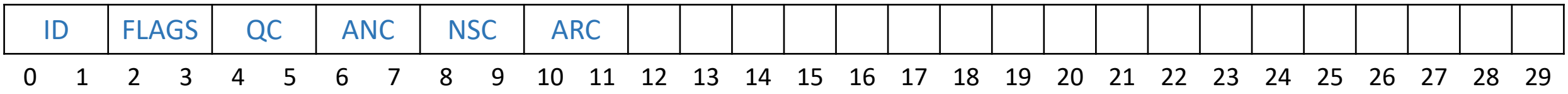
Parsing Names (no pointers)

Binary

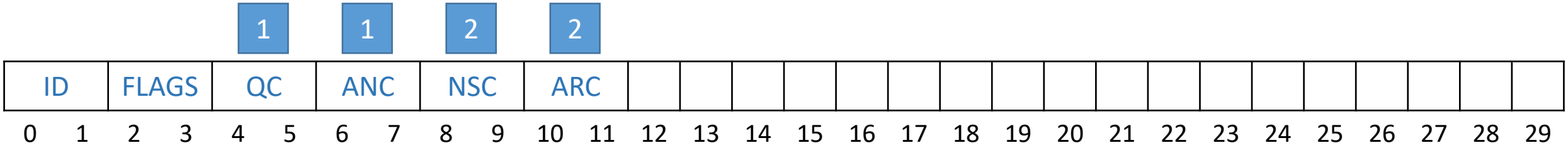
ASCII



Parsing Names (no pointers)



Parsing Names (no pointers)

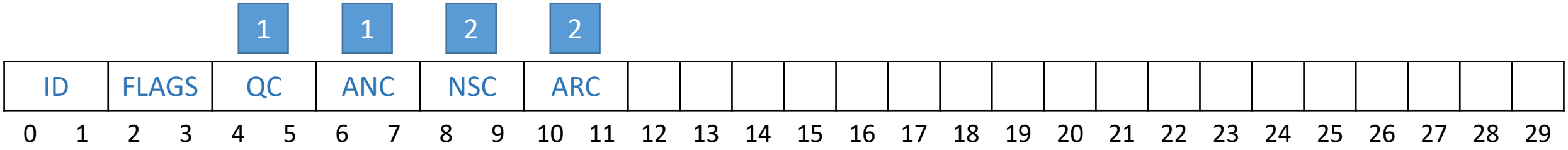



```
parse_question(response, 12)
```

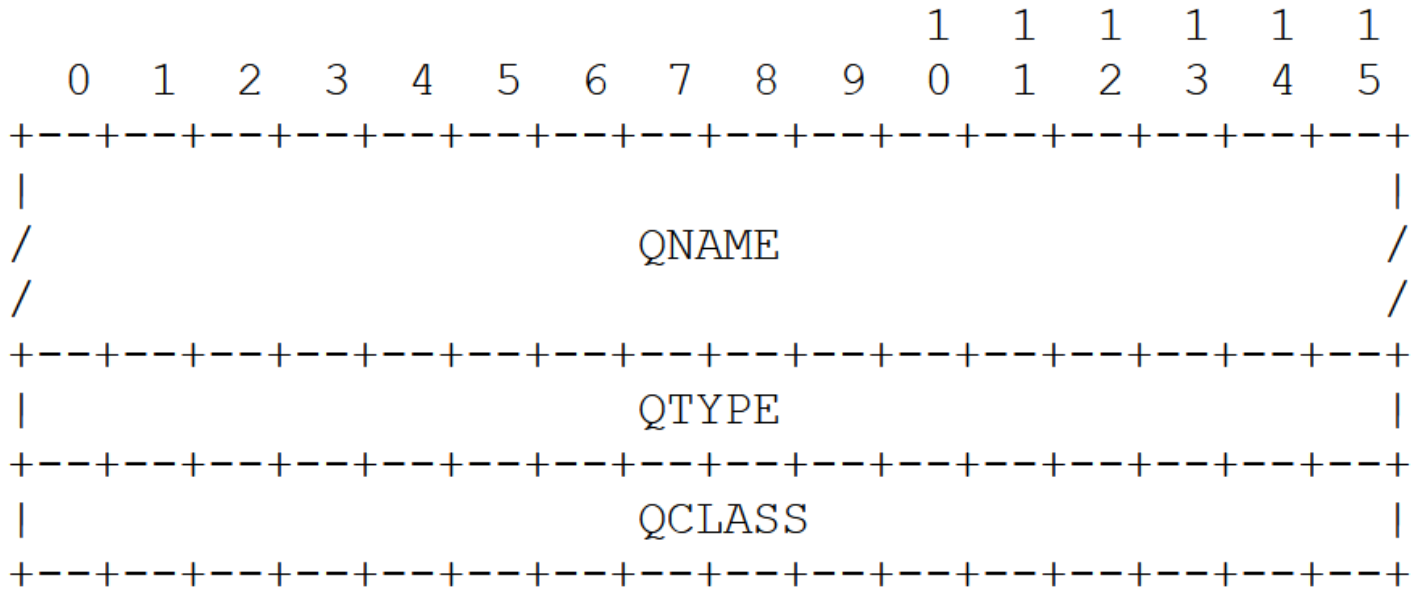
Parsing Names (no pointers)

Binary

ASCII




```
parse_question(response, 12)

  parse_name(response, 12)
```

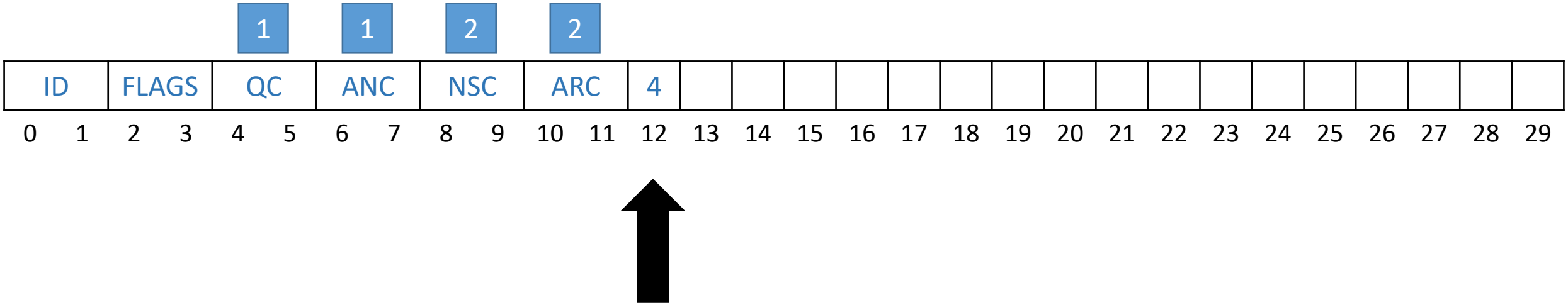


Name so far:

Parsing Names (no pointers)

Binary

ASCII



Unpack one-byte, unsigned integer: !B

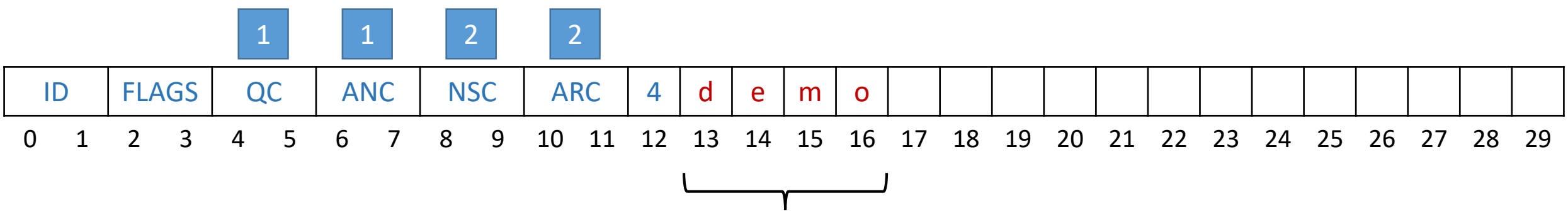
Does this byte have the two high order bits set to 1?

If yes, pointer
This case: no, not a pointer

Is it 0? If so, end of name

Name so far:

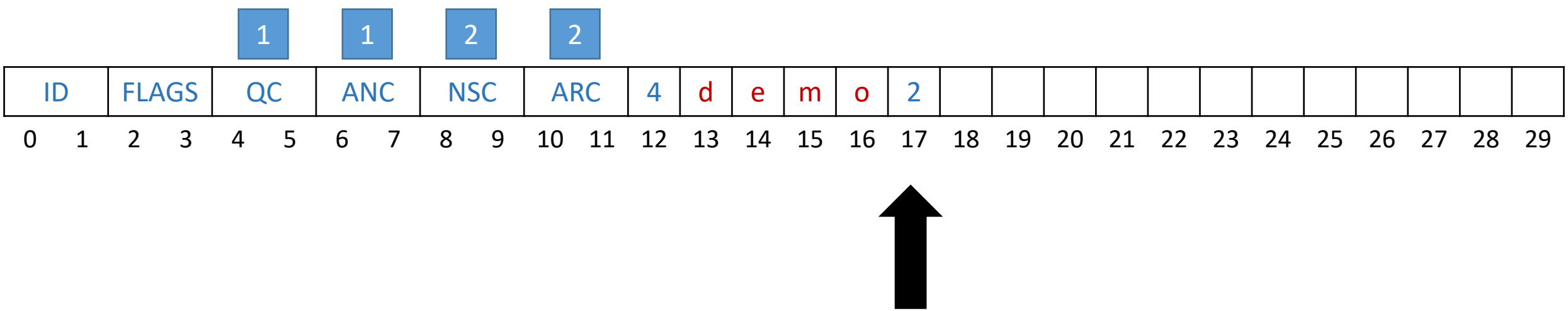
Parsing Names (no pointers)



Next 4 characters are one "label" or chunk of the name.

Name so far: demo

Parsing Names (no pointers)



Unpack one-byte, unsigned integer: !B

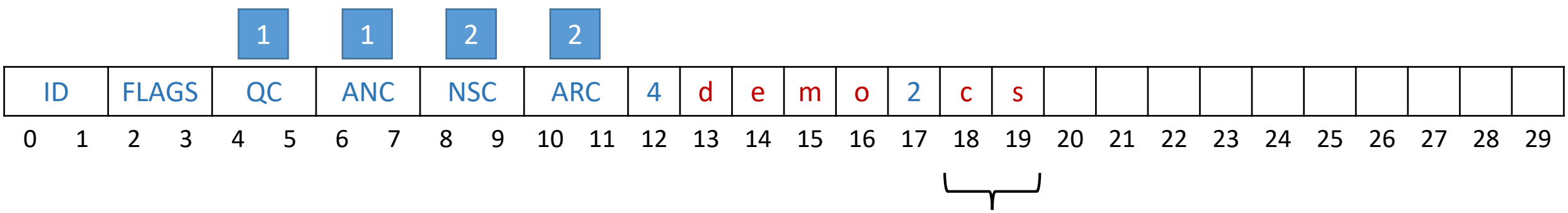
Does this byte have the two high order bits set to 1?

If yes, pointer
This case: no, not a pointer

Is it 0? If so, end of name

Name so far: demo

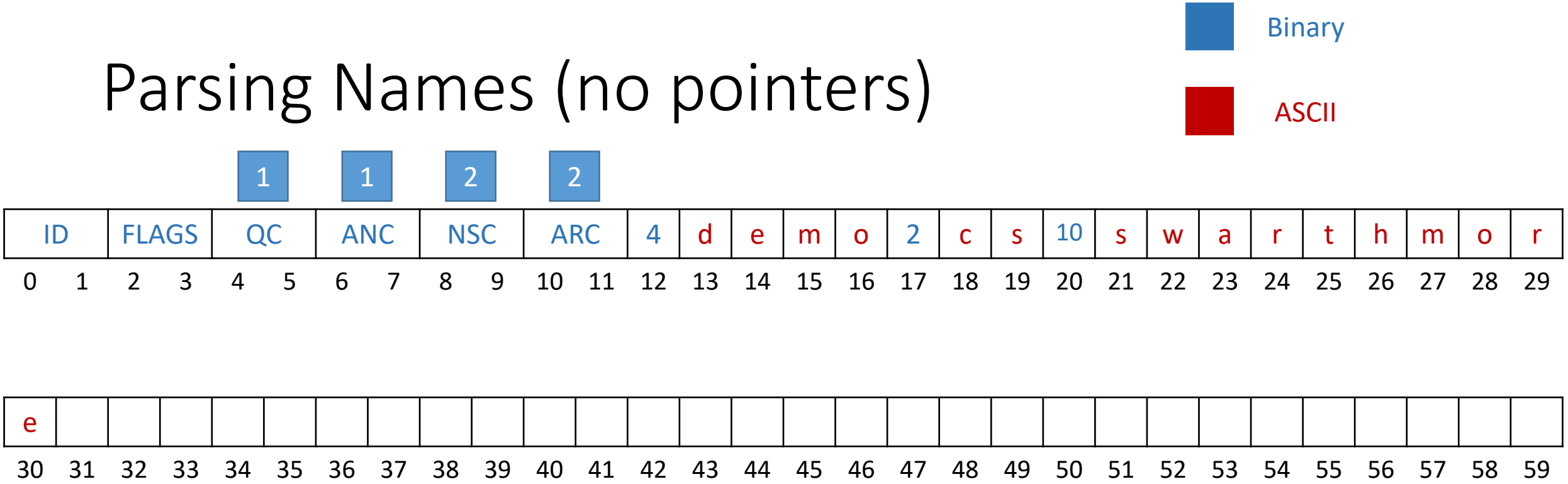
Parsing Names (no pointers)



Next 2 characters are one "label" or chunk of the name.

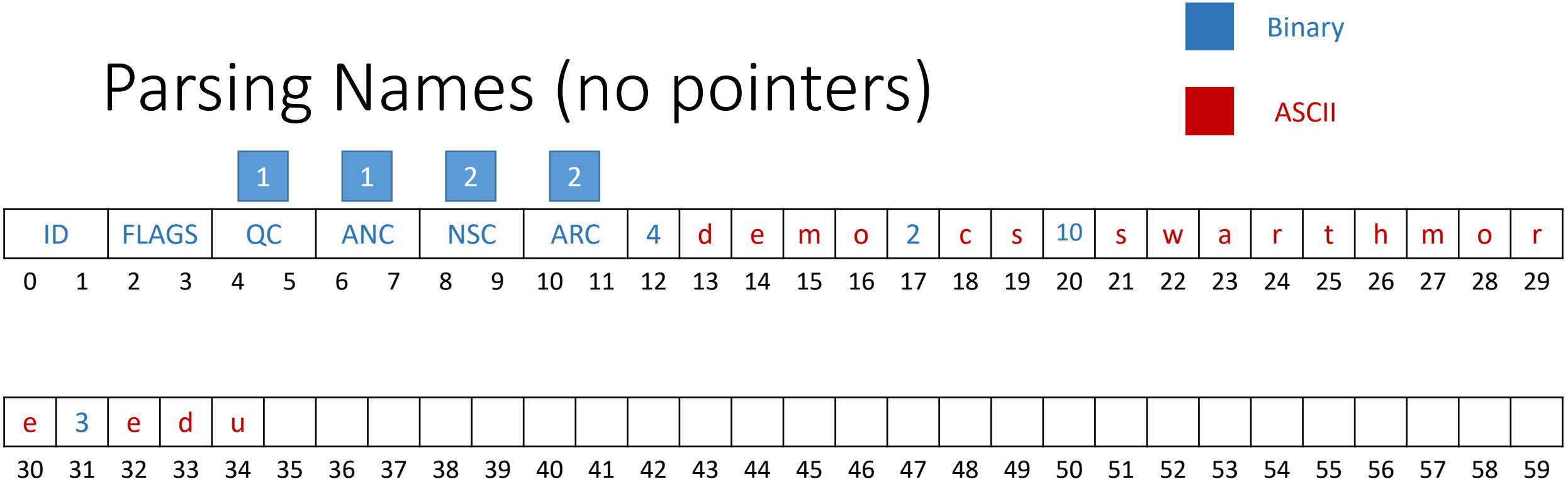
Name so far: demo.cs

Parsing Names (no pointers)



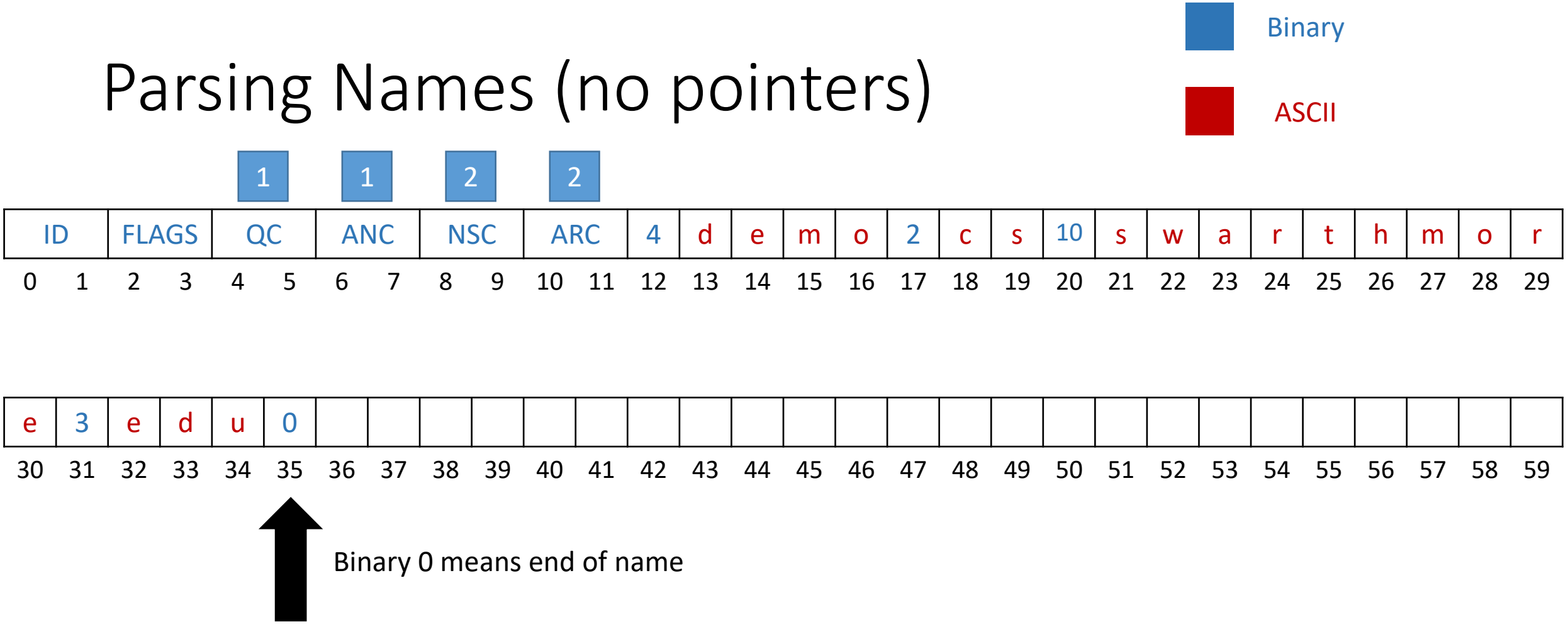
Name so far: demo.cs.swarthmore

Parsing Names (no pointers)



Name so far: demo.cs.swarthmore.edu

Parsing Names (no pointers)



Name so far: demo.cs.swarthmore.edu

Parsing Names (no pointers)

				1		1		2		2																			
ID		FLAGS		QC		ANC		NSC		ARC		4	d	e	m	o	2	c	s	10	s	w	a	r	t	h	m	o	r
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

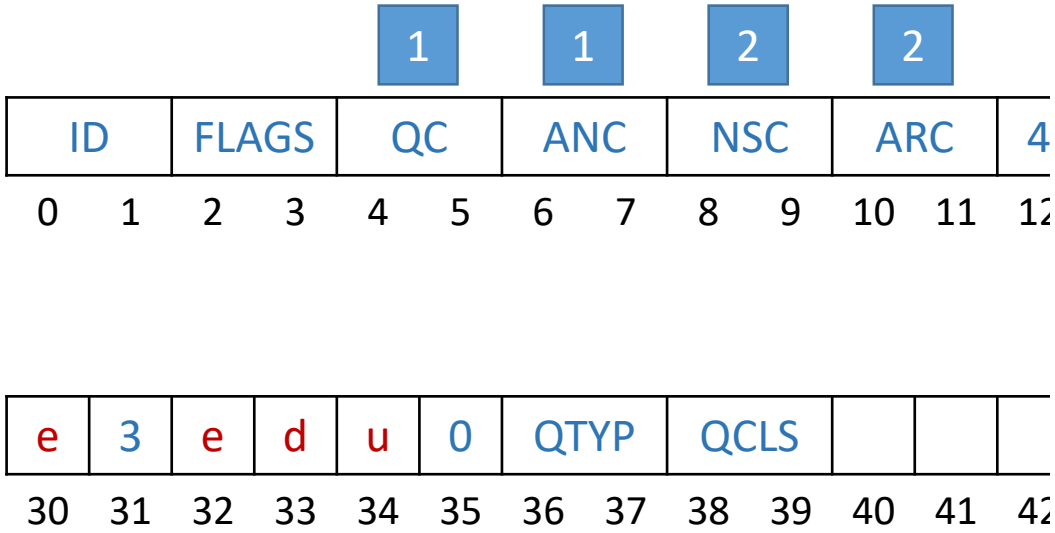
e	3	e	d	u	0																								
30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59

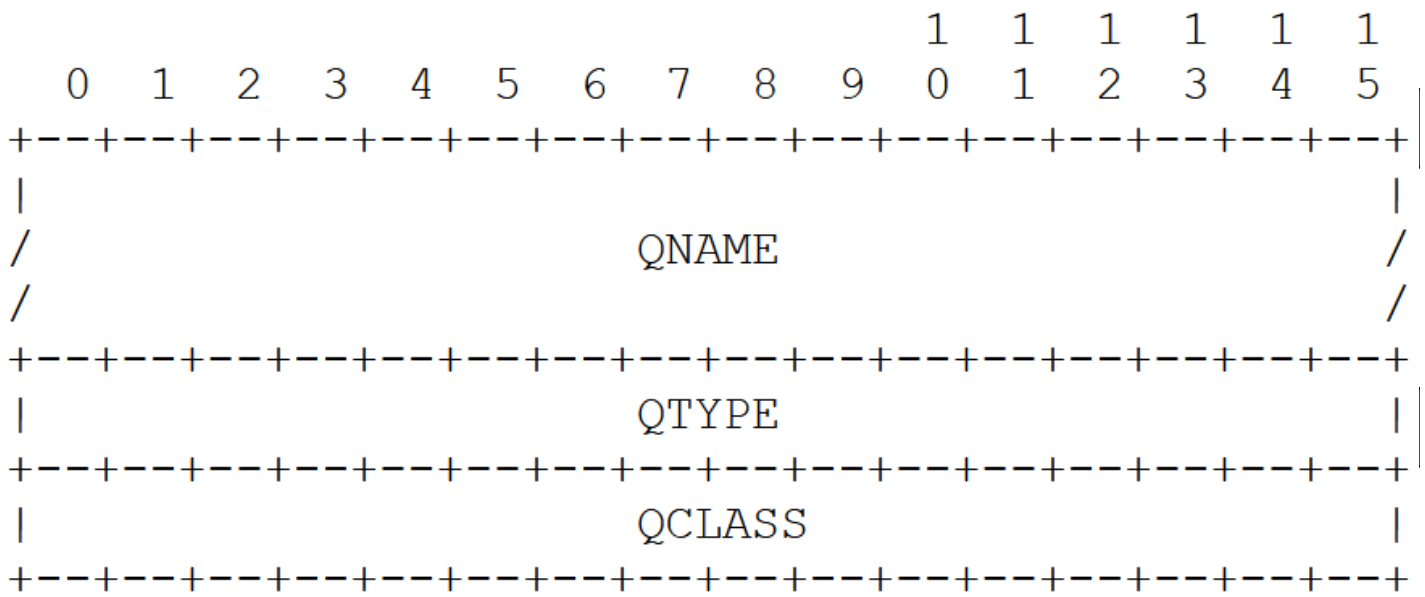
```
parse_name(response, 12)  =>  ('demo.cs.swarthmore.edu', 24)
```

Parsing Names (no pointers)

Binary

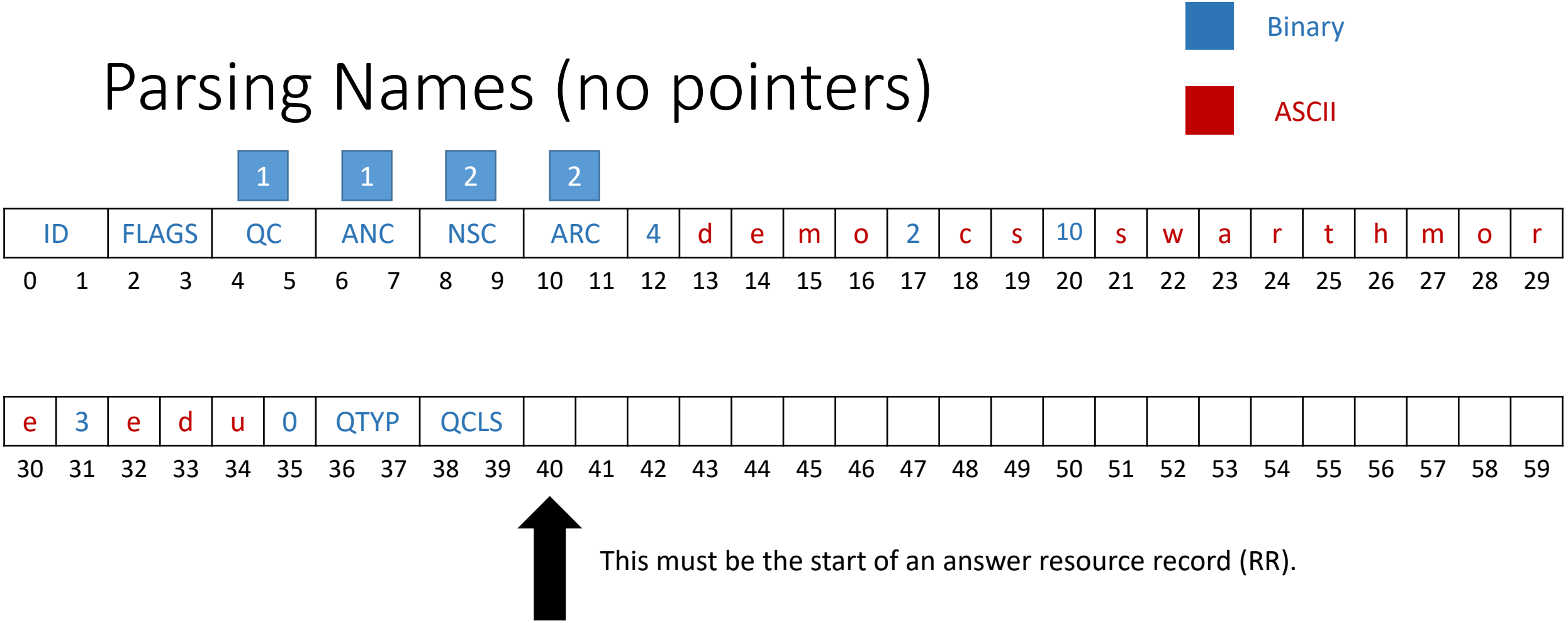
ASCII






```
parse_name(response, 12)  =>  ('demo.cs.swarthmore.edu', 24)

parse_question(response, 12)  => 28
```

Parsing Names (no pointers)



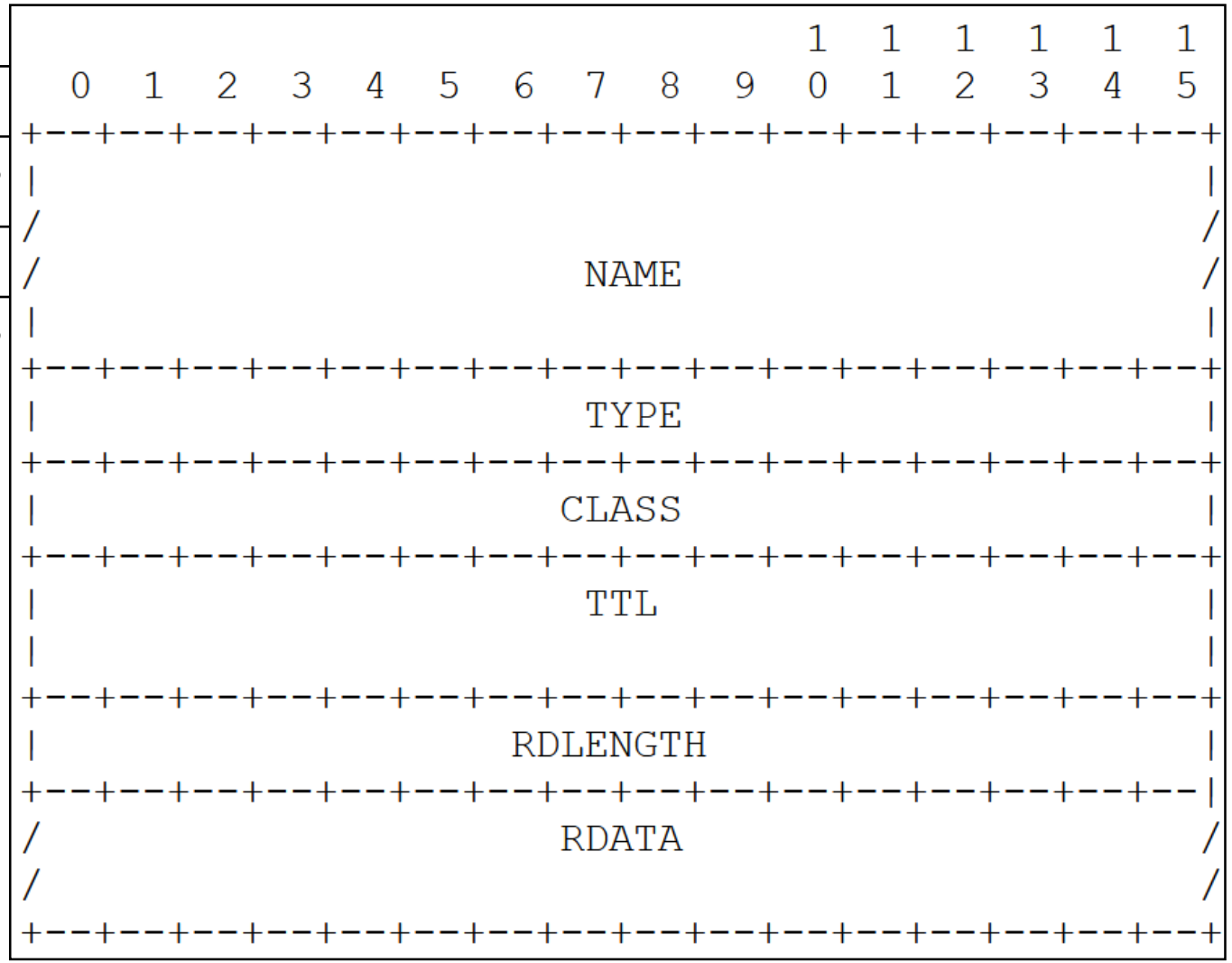
Parsing Names (with pointers)

Binary

ASCII

						1		1		2		2	
ID		FLAGS		QC		ANC		NSC		ARC		4	d
0	1	2	3	4	5	6	7	8	9	10	11	12	13
e	3	e	d	u	0	QTYP		QCLS					
30	31	32	33	34	35	36	37	38	39	40	41	42	43

```
parse_resource(response, 40)
```



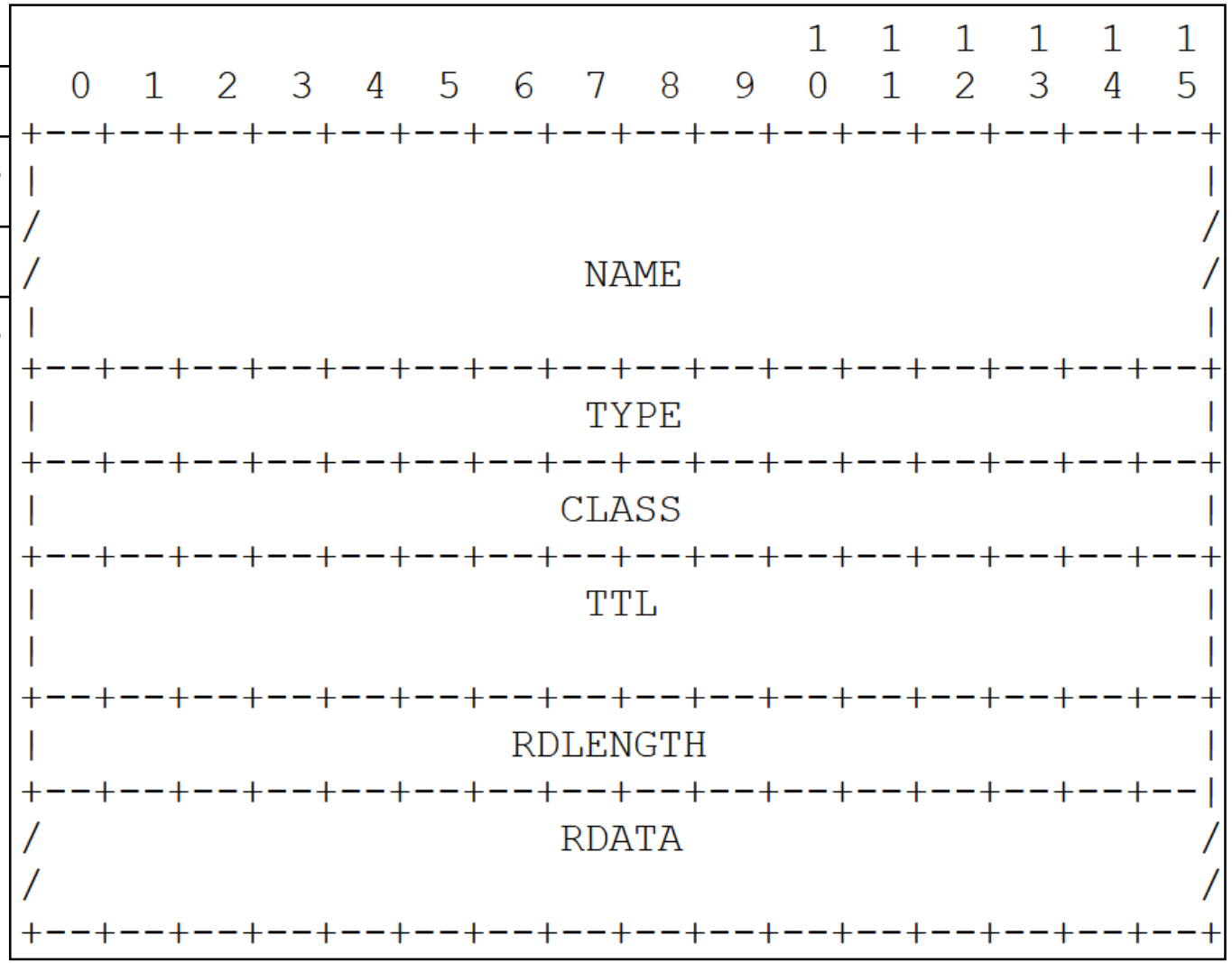
Parsing Names (with pointers)

Binary

ASCII

						1		1		2		2	
ID		FLAGS		QC		ANC		NSC		ARC		4	d
0	1	2	3	4	5	6	7	8	9	10	11	12	13
e	3	e	d	u	0	QTYP		QCLS					
30	31	32	33	34	35	36	37	38	39	40	41	42	43

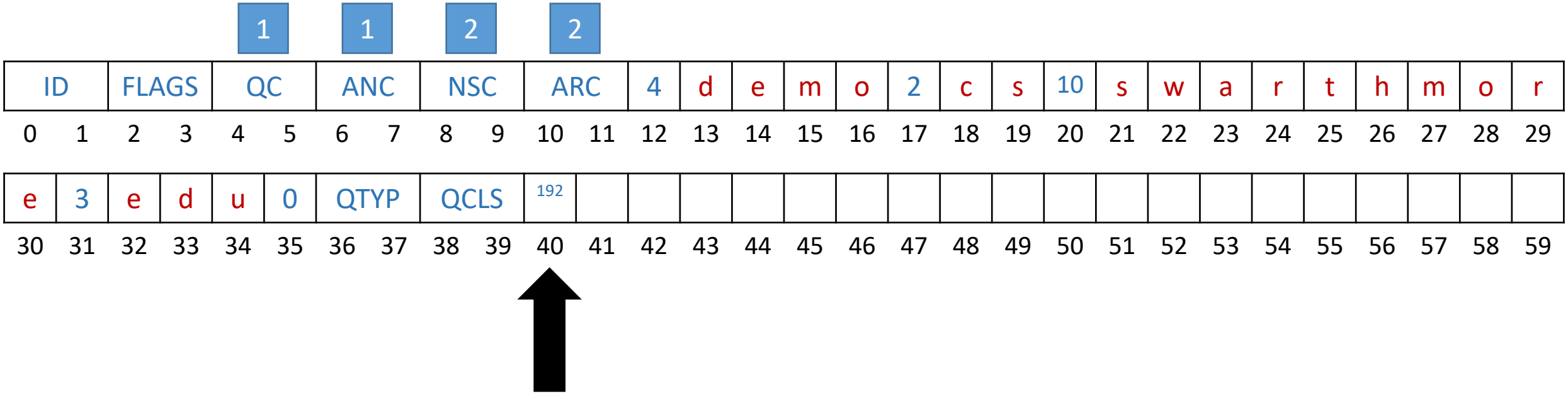
```
parse_resource(response, 40)

  parse_name(response, 40)
```



Parsing Names (with pointers)

Binary

ASCII

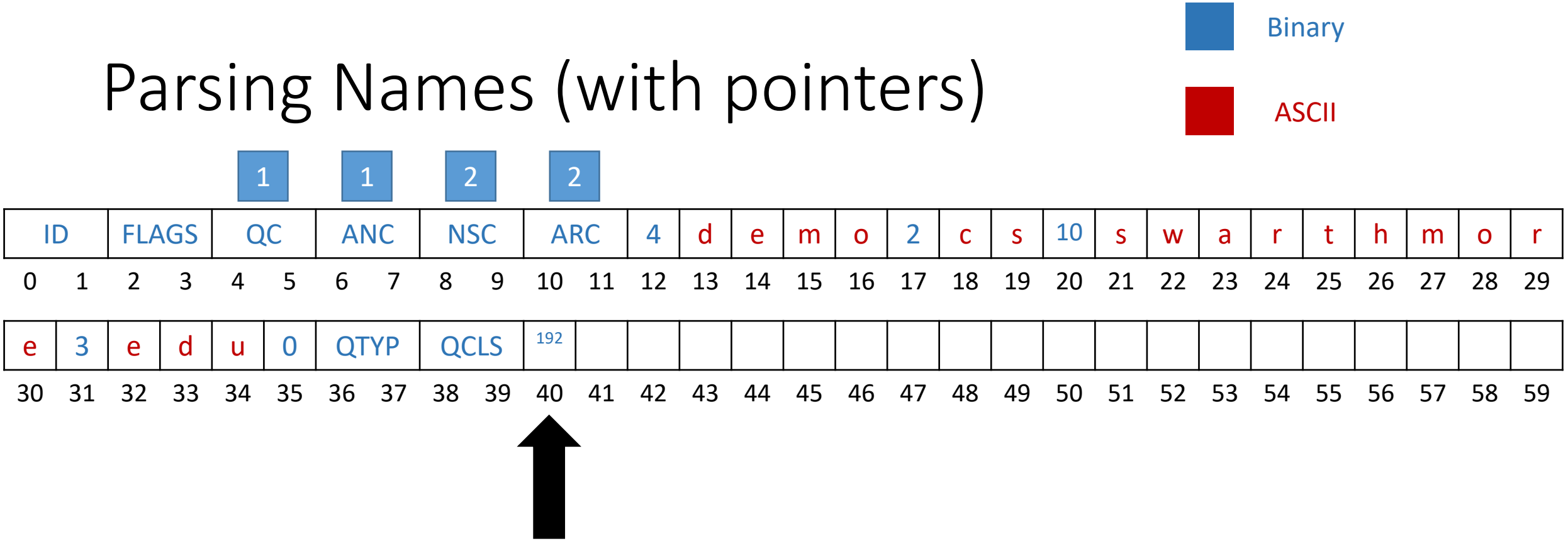


Unpack one-byte, unsigned integer: !B

Does this byte have the two high order bits set to 1?

Name so far:

Parsing Names (with pointers)



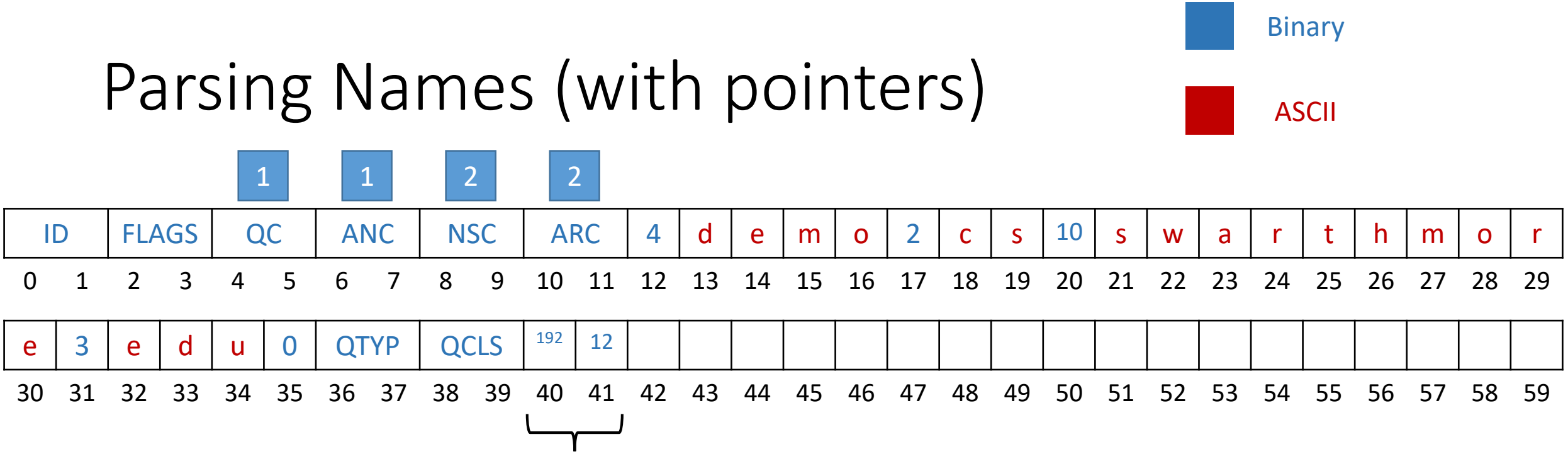
Unpack one-byte, unsigned integer: !B

Does this byte have the two high order bits set to 1?

192 in binary: `11000000`

Name so far:

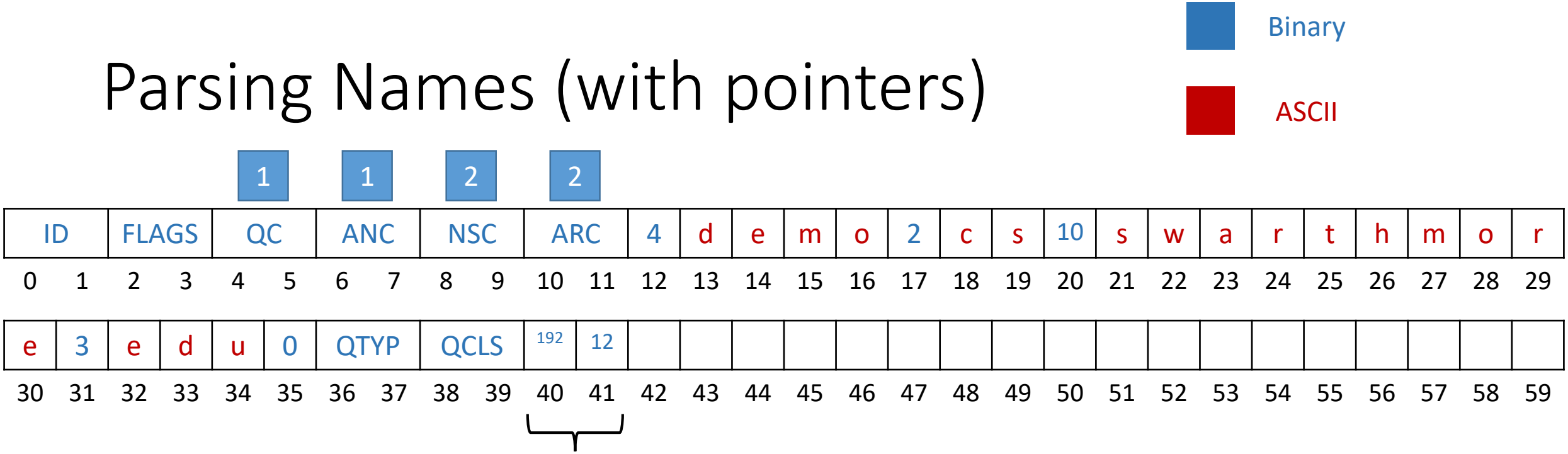
Parsing Names (with pointers)



Take the first type (with high-order 11 bits) and the next byte and treat them as a two-byte value: unpack them with !H

Name so far:

Parsing Names (with pointers)



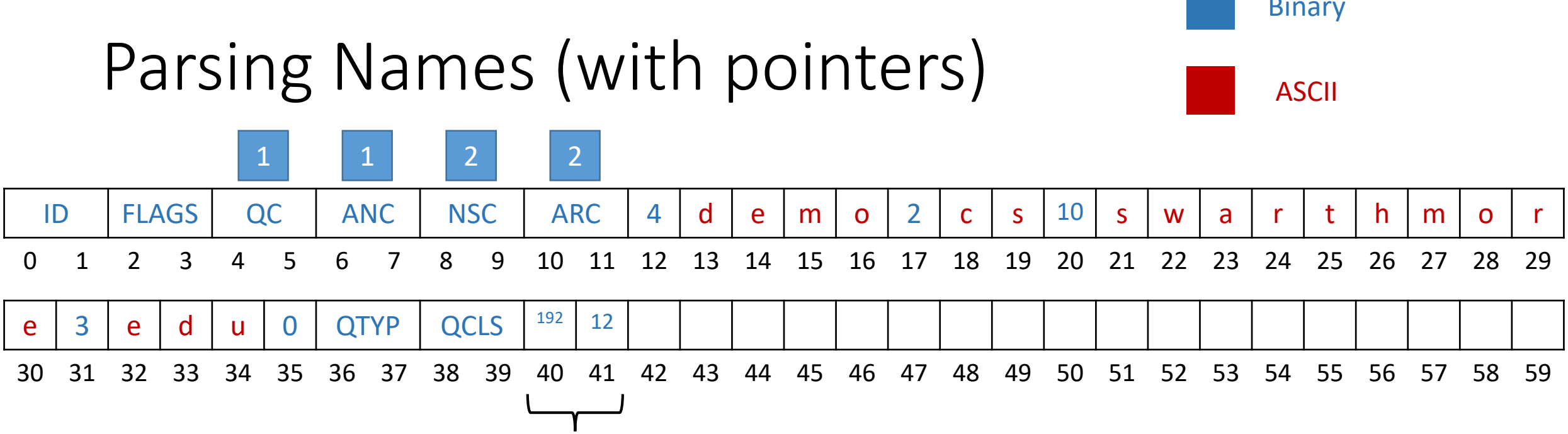
Take the first type (with high-order 11 bits) and the next byte and treat them as a two-byte value: unpack them with !H

Together, these bytes are: `11000000 00001100`

Because the first two bits indicate "pointer", we ignore them: ~~`00`~~`000000 00001100`

Name so far:

Parsing Names (with pointers)

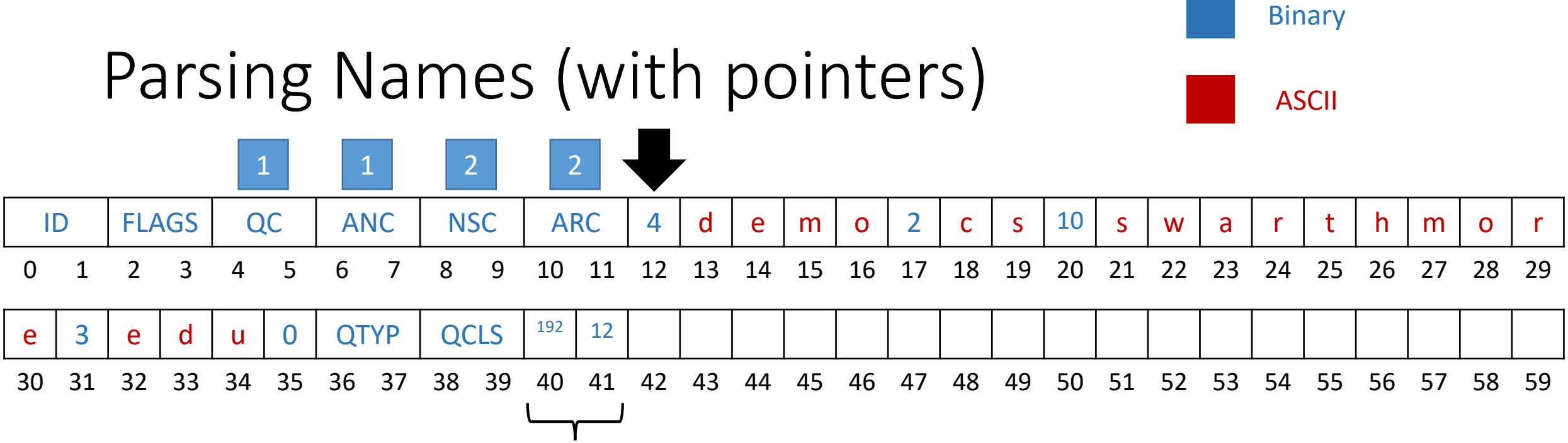


Because the first two bits indicate "pointer", we ignore them: ~~00~~000000 00001100

Thus, the pointer is pointing to offset 12, which means "resume building the name you're looking for at offset 12 of the message".

Name so far:

Parsing Names (with pointers)

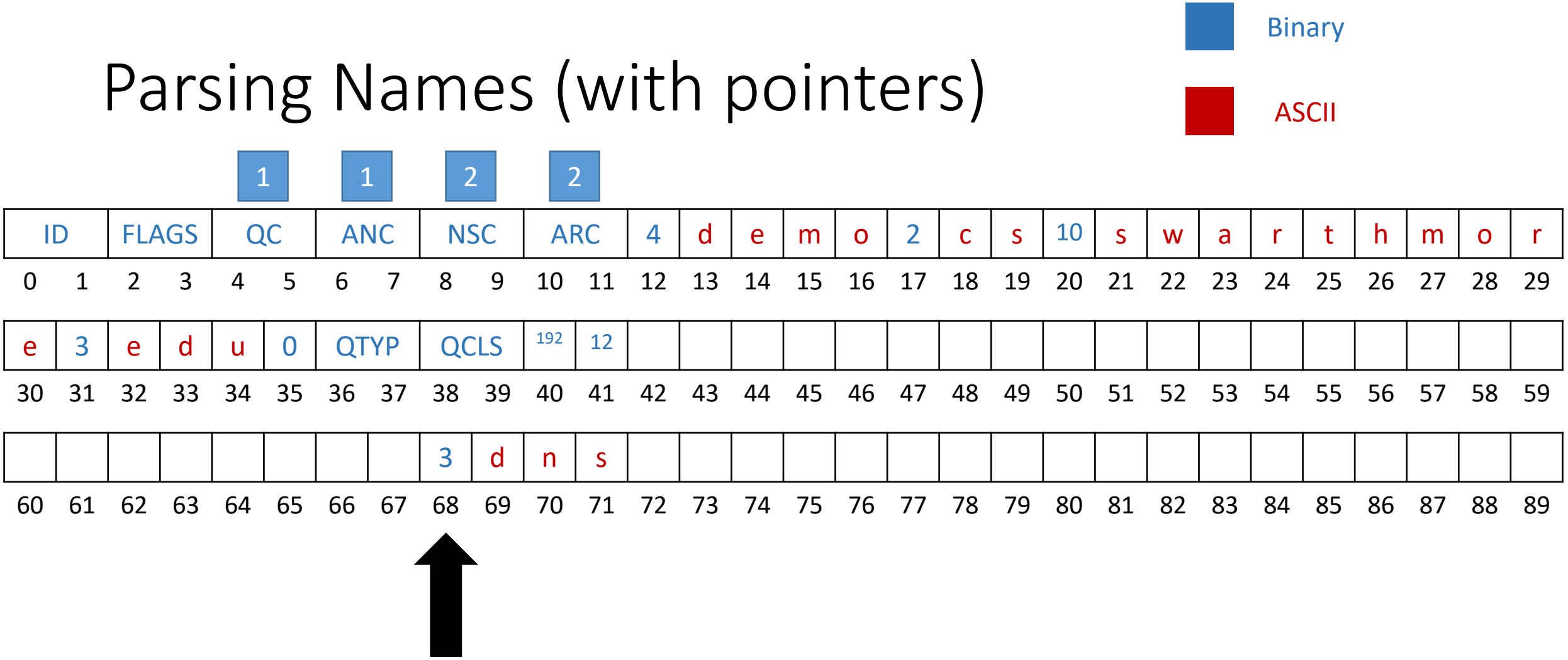


Because the first two bits indicate "pointer", we ignore them: ~~00~~000000 00001100

Thus, the pointer is pointing to offset 12, which means "resume building the name you're looking for at offset 12 of the message".

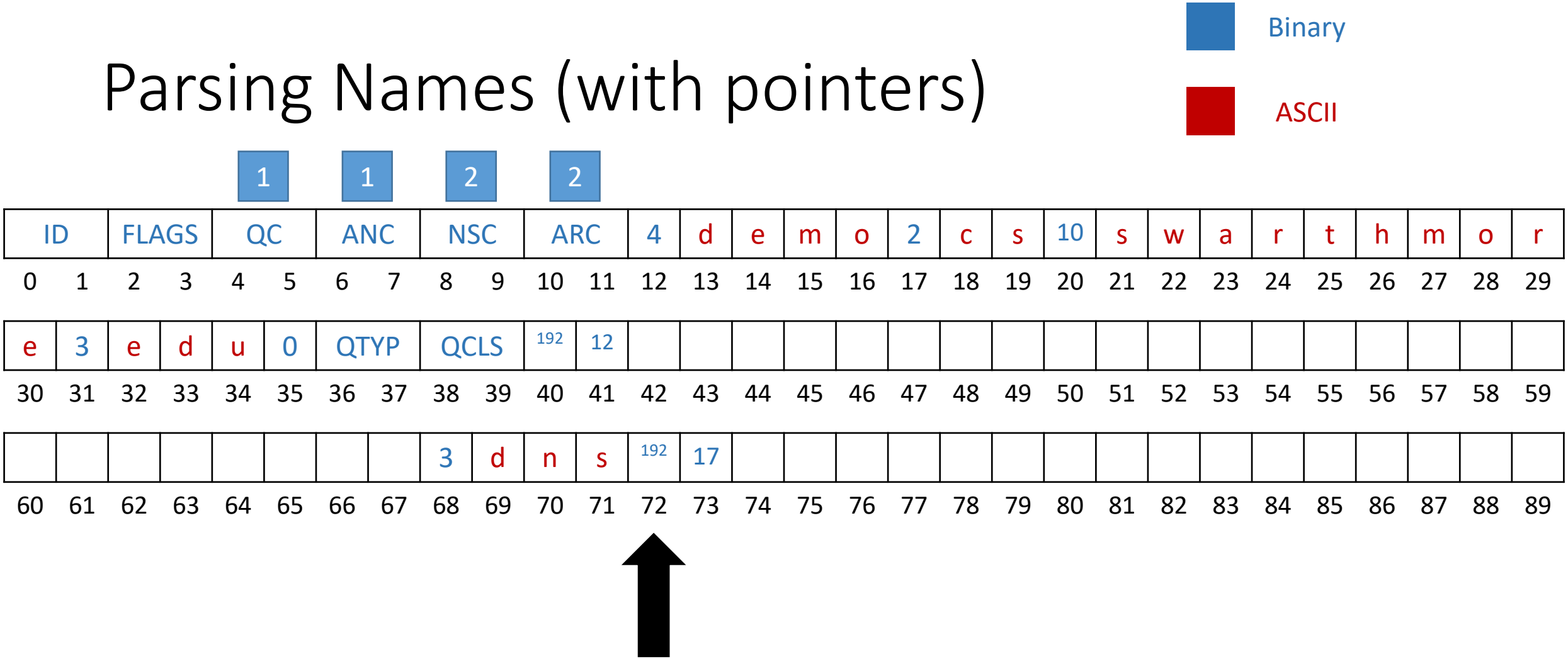
Name so far:

Parsing Names (with pointers)



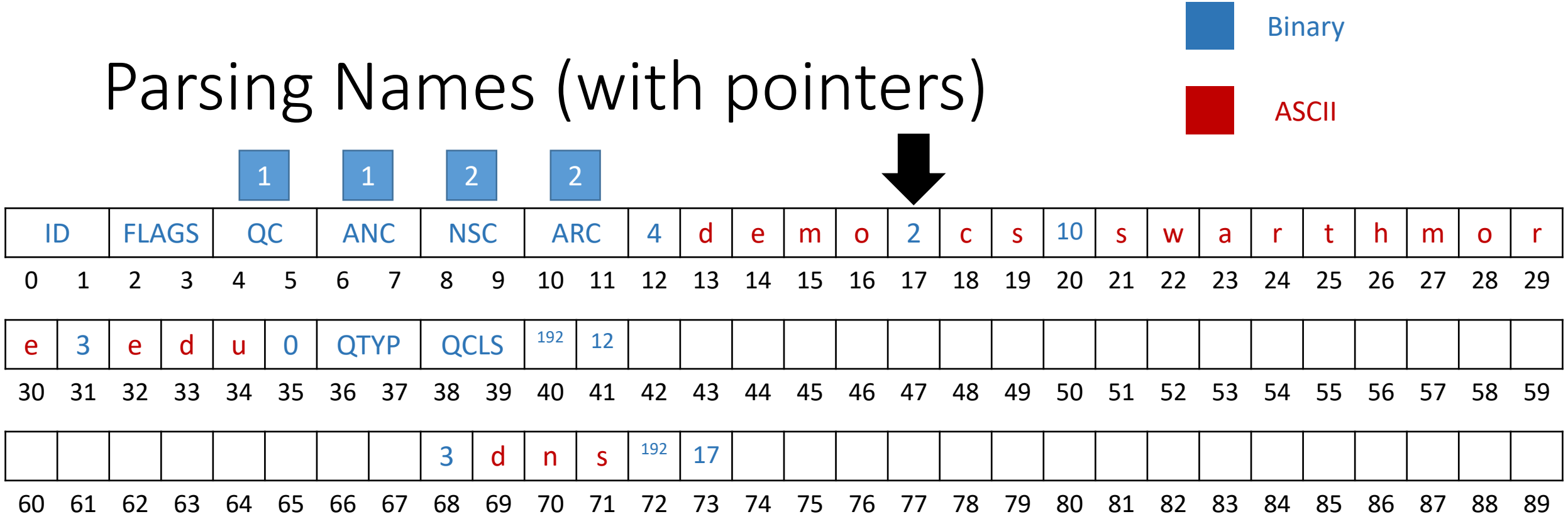
Name so far: dns

Parsing Names (with pointers)



Name so far: dns

Parsing Names (with pointers)



Name so far: dns + (whatever we find at offset 17) => dns.cs.swarthmore.edu