

Type Cast (1A)

Copyright (c) 2010-2018 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using LibreOffice.

Type Conversion

Implicit Type Conversion

Promotion

`1 / 2.0` → `1.0 / 2.0`

Demotion

`int m;`
`m = 1.25;` → `m = 1;` truncation problem

Explicit Type Conversion

`m = (int) 1.25;` No warning message

Type Conversion (int → double) - implicit one

Implicit Type Conversion

Promotion

```
int m;  
m = 123;
```



```
double x;  
x = m;
```

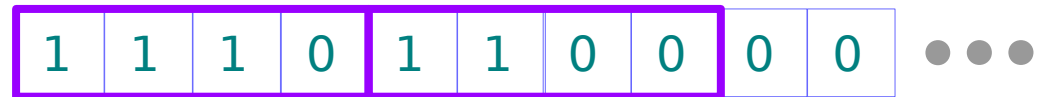
0x7B



$(0x7B / 2^6) * 2^6$



implicit one



0xEC 00 00 00 ...

Type Conversion (int → double) - 16 hexa digits

Fraction : 52bit = 4 * 13 → **13** hexa digits

0xEC 00 00 00 00 00 0

Exponent : 11bit

Sign : 1bit = 4 * 3 → **3** hexa digits

$(0x7B / 2^6) * 2^6$


excess 1023 code

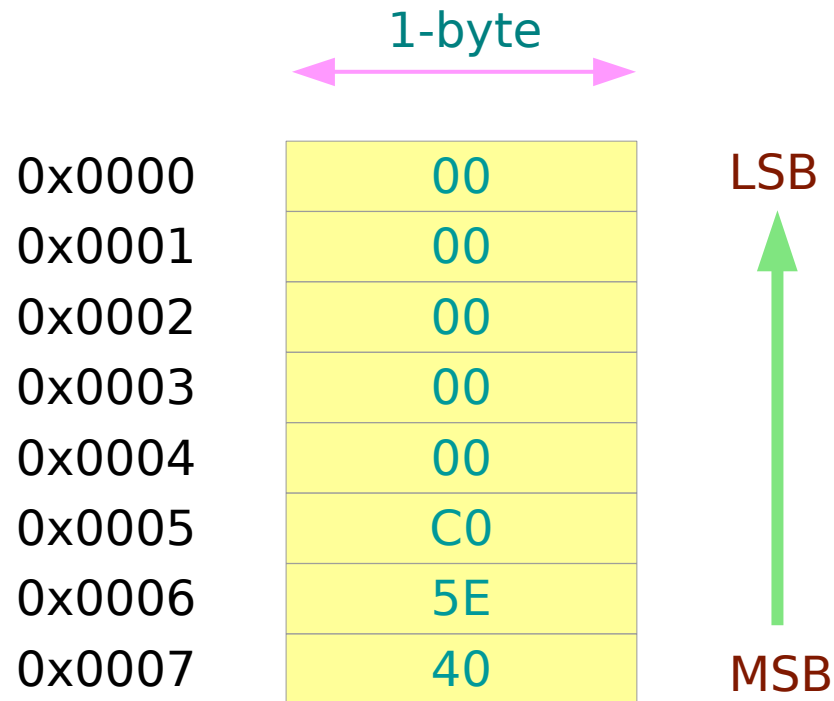
$6 + 1023 = 1029 \rightarrow 0x405$

0x405 EC 00 00 00 00 00 0

0x40 5E C0 00 00 00 00 00

Type Conversion (int → double) - Little Endian

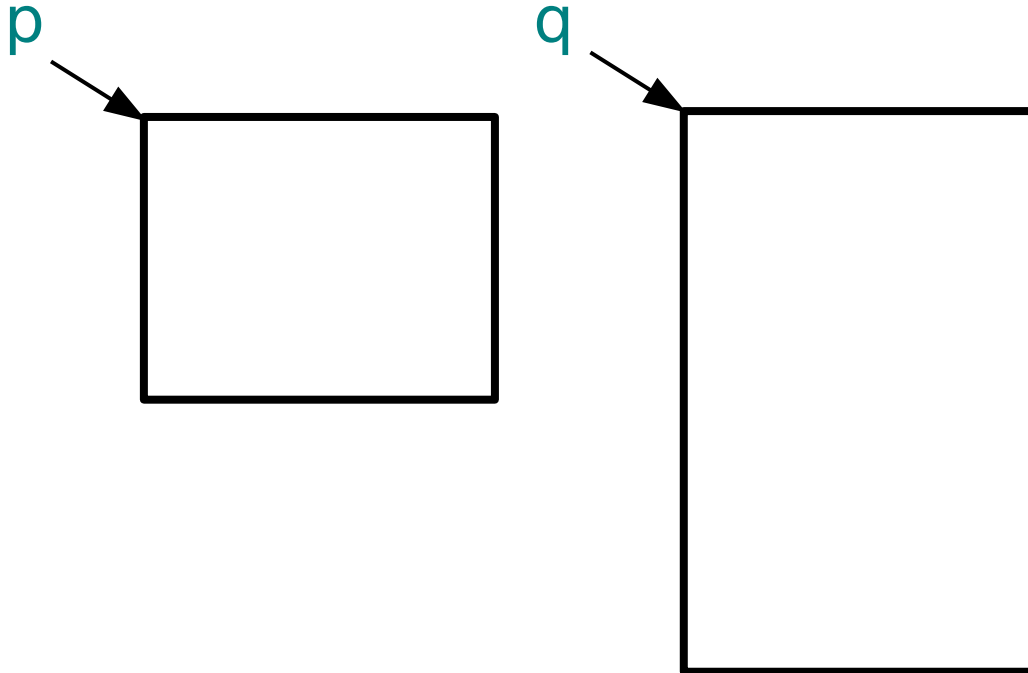
MSB  LSB
0x40 5E C0 00 00 00 00 00



Little Endian:
LSByte First

Type Casting

```
double x = 123;  
int *p;  
double *q;
```



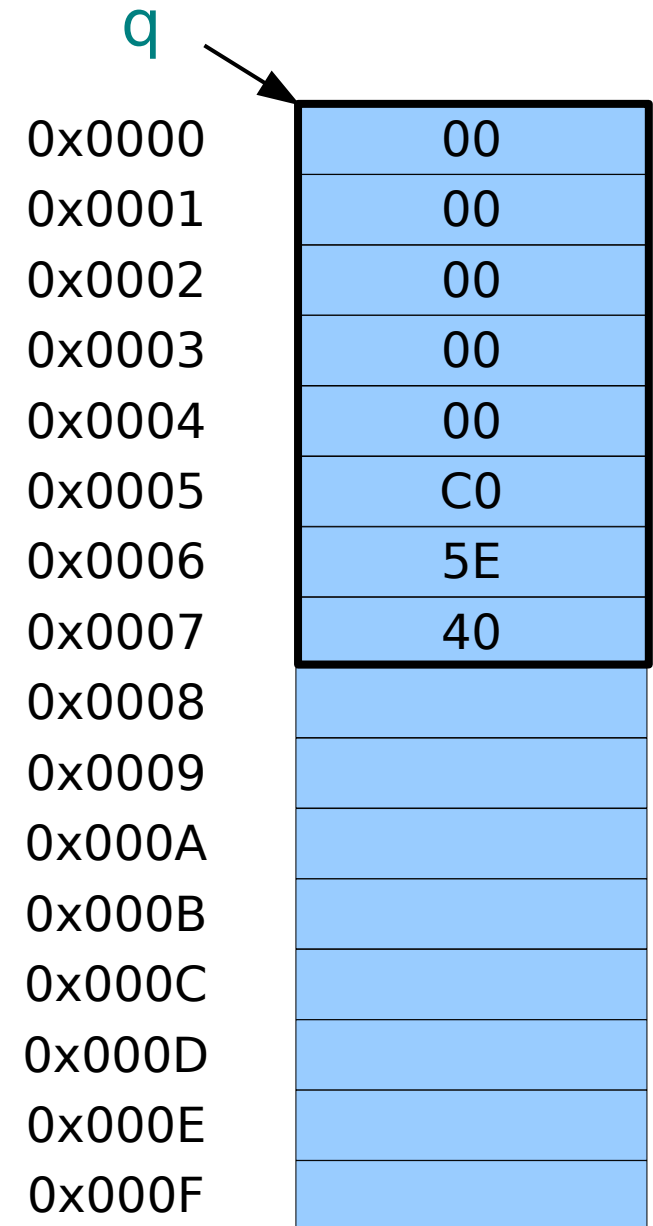
0x0000	00
0x0001	00
0x0002	00
0x0003	00
0x0004	00
0x0005	C0
0x0006	5E
0x0007	40
0x0008	
0x0009	
0x000A	
0x000B	
0x000C	
0x000D	
0x000E	
0x000F	

Type Casting

```
double x = 123;  
int     *p;  
double  *q;
```

```
q = &x;
```

```
*q → 123.0
```

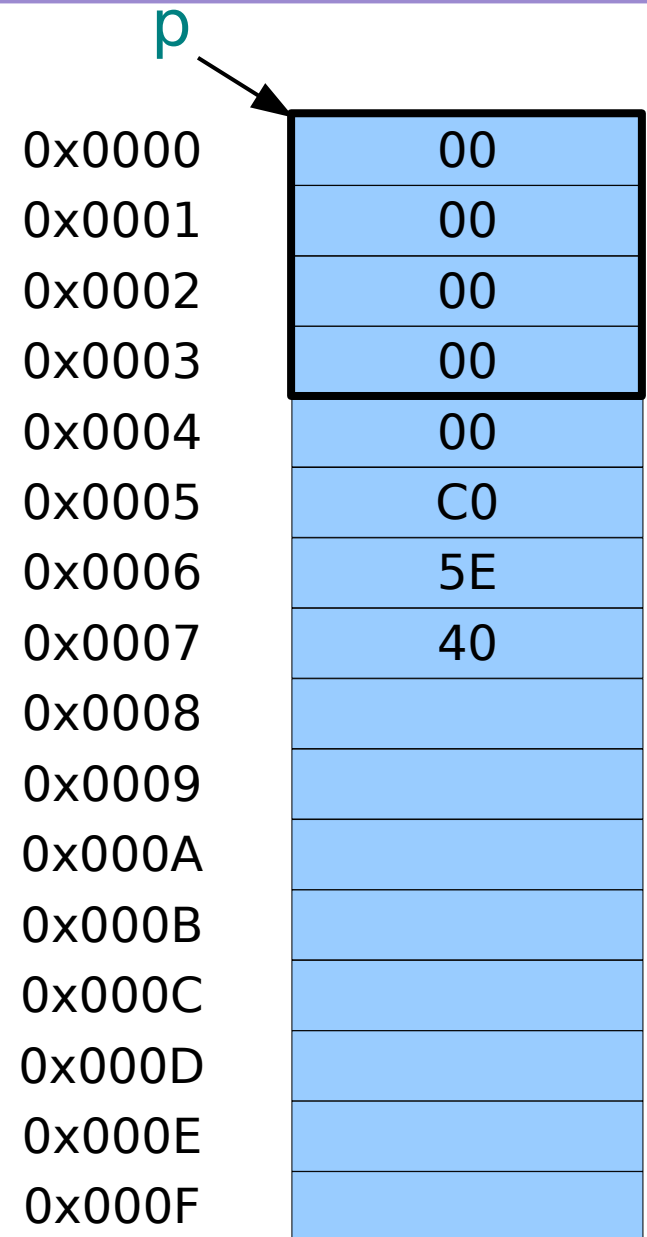


Type Casting

```
double x = 123;  
int     *p;  
double  *q;
```

```
p = (int *) &x;
```

```
*p → 0
```



Pointer Type Cast

Changing the associated data type of an address

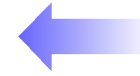
long a;

address of a long value



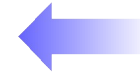
int * p;

address of an int value



short* q;

address of a short value



char * r;

address of a char value



Pointer Type Casting

long a;

address of a long value

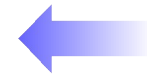


&a



int * p;

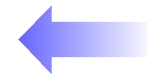
address of an int value



p = (int *) &a

short * q;

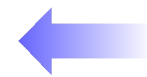
address of a short value



q = (short *) &a

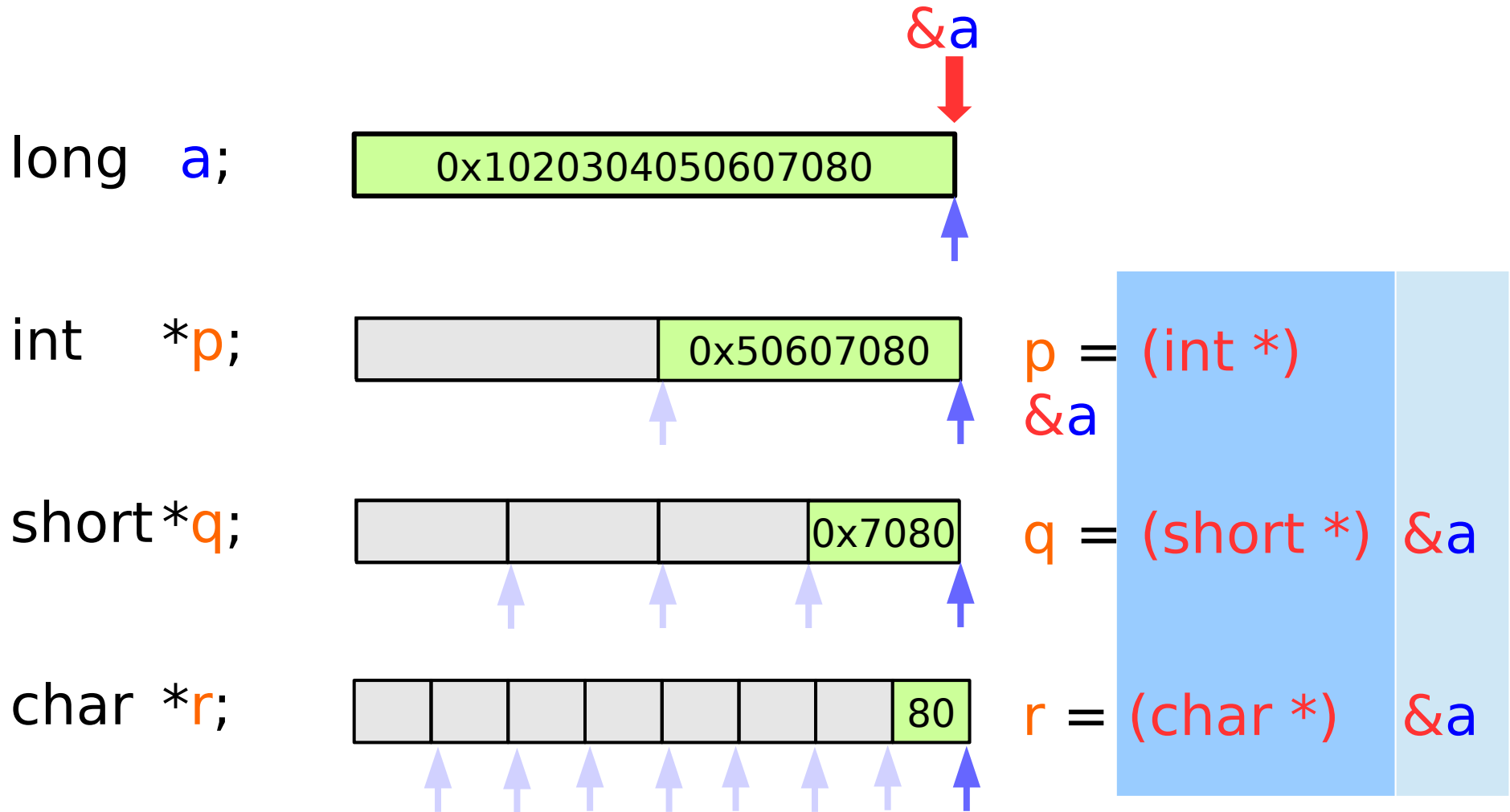
char * r;

address of a char value



r = (char *) &a

Re-interpretation of memory data - case I



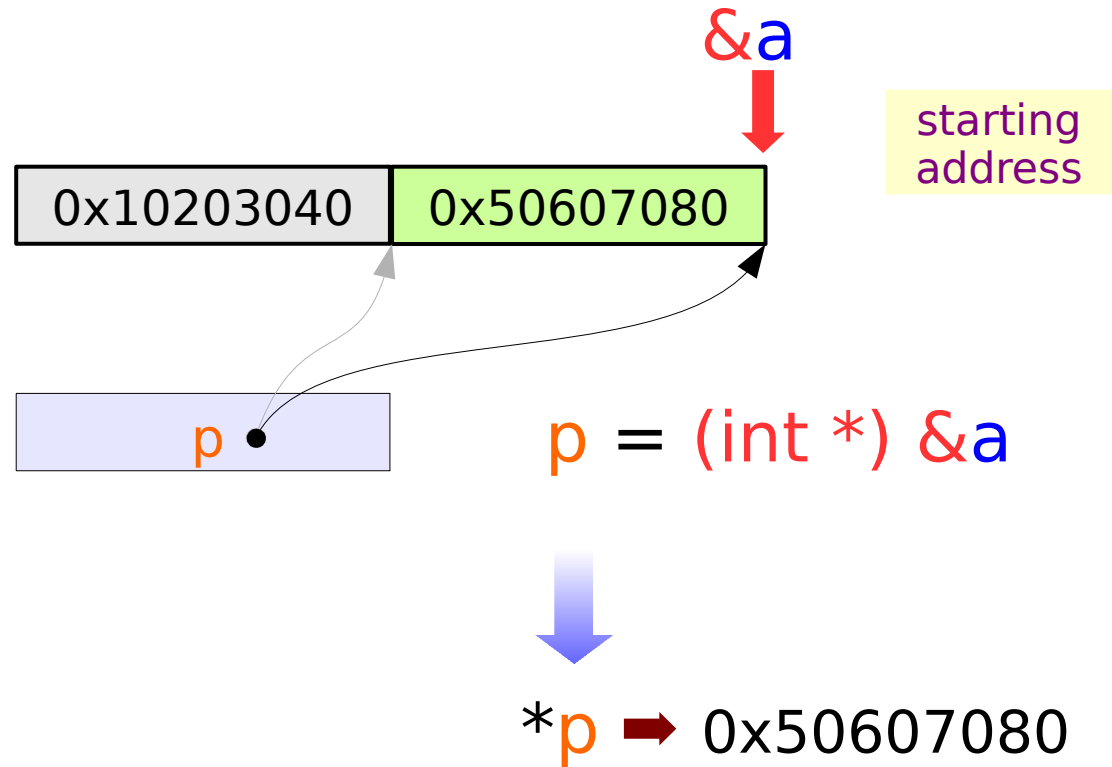
Pointer Type Cast

long a;

int *p;

short *q;

char *r;



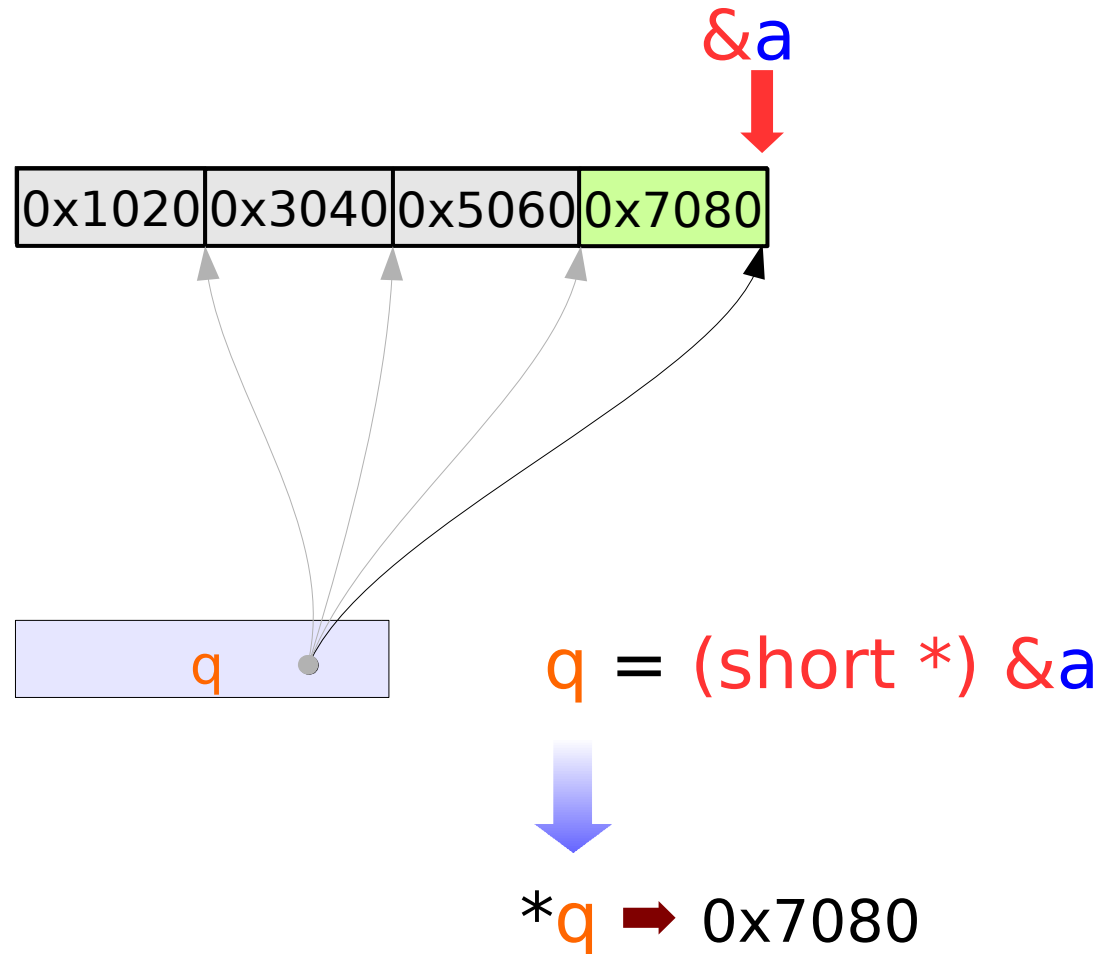
Integer Pointer Types

long a;

int *p;

short *q;

char *r;



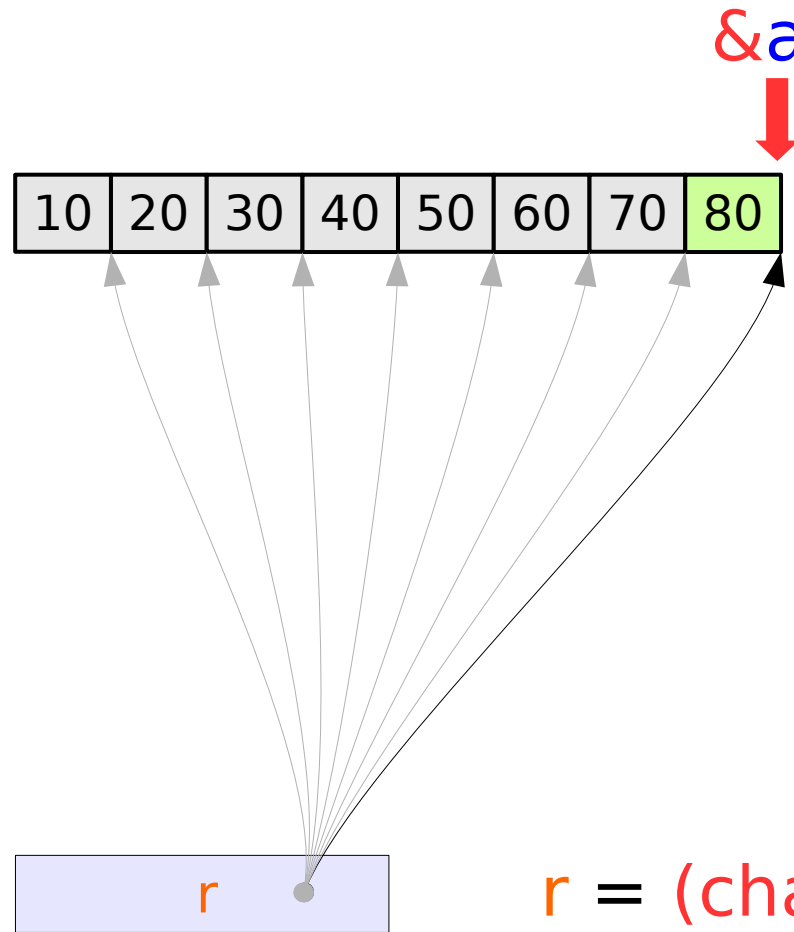
Integer Pointer Types

long `a`;

int `*p`;

short `*q`;

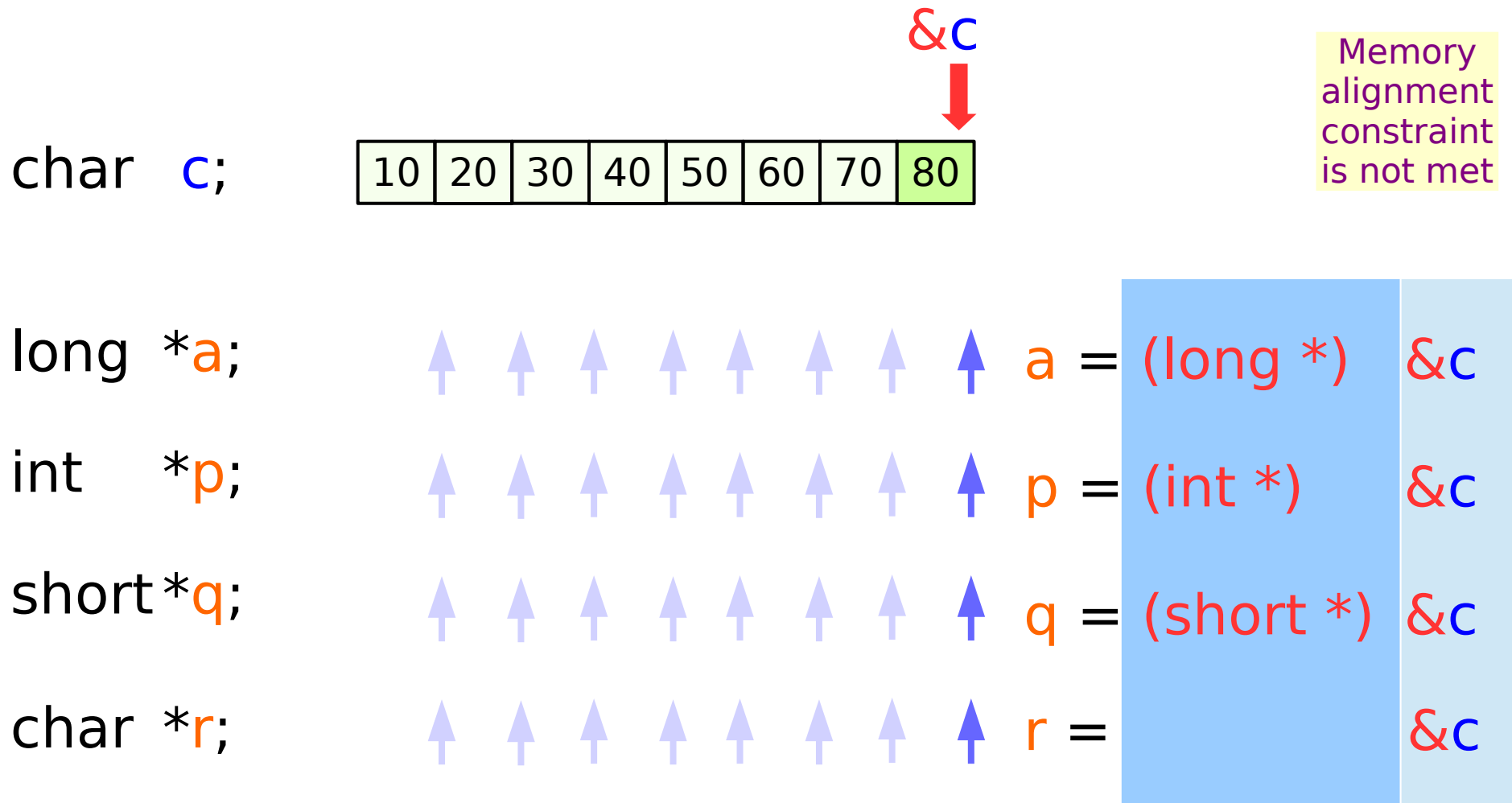
char `*r`;



`r = (char *) &a`

`*r` \rightarrow 0x80

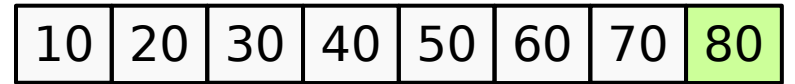
Re-interpretation of memory data – case II



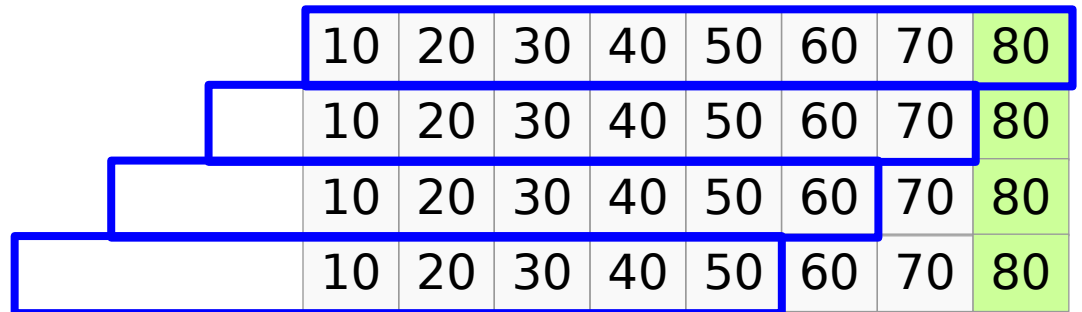
In this case, the memory alignment constraint can be broken

Pointer Type Cast

char c;

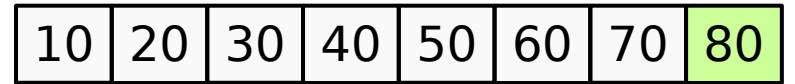


long *a;

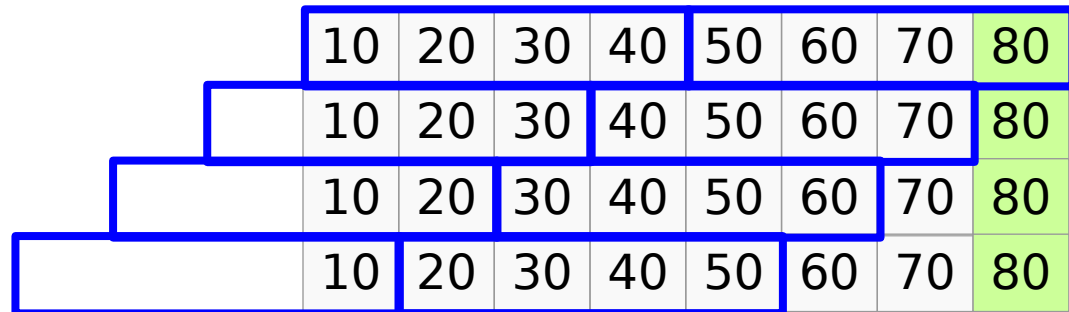


Pointer Type Cast

char c;

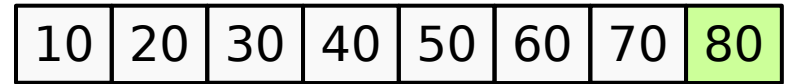


int *p;

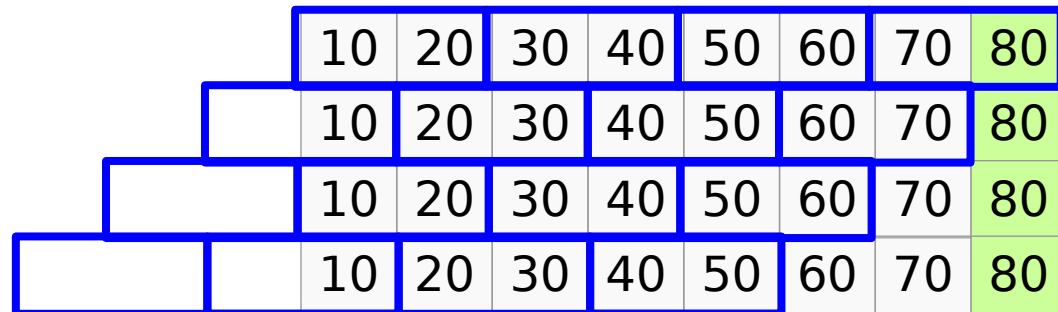


Pointer Type Cast

char c;



short*q;



References

- [1] Essential C, Nick Parlante
- [2] Efficient C Programming, Mark A. Weiss
- [3] C A Reference Manual, Samuel P. Harbison & Guy L. Steele Jr.
- [4] C Language Express, I. K. Chun
- [5] <http://www.stackoverflow.com>