

Computer System Organization

Recitation

[Fall 2018]

CSCI-UA 201-006

R2: C variable/pointer/array/bit operators

Variable in C

- Note that the definition here is not formal but one way to understand C language.
- A variable defines three information.
 - The memory address(ADDR) of that variable.
 - The number of bytes used by the variable.
 - How to interpret the content stored in ADDR.

Variable in C

char c = 'A';

***(0x1000F000)
= 'A'**

Address	Disassembly	Comment	Value
00401000	int c;		00000000
00401004	char c;		00000000
00401008	int a;		00000000
0040100C	float b;		00000000
00401010
00401014
00401018
0040101C
00401020
00401024
00401028
0040102C
00401030
00401034
00401038
0040103C
00401040
00401044
00401048
0040104C
00401050
00401054
00401058
0040105C
00401060
00401064
00401068
0040106C
00401070
00401074
00401078
0040107C
00401080
00401084
00401088
0040108C
00401090
00401094
00401098
0040109C
004010A0
004010A4
004010A8
004010AC
004010B0
004010B4
004010B8
004010BC
004010C0
004010C4
004010C8
004010CC
004010D0
004010D4
004010D8
004010DC
004010E0
004010E4
004010E8
004010EC
004010F0
004010F4
004010F8
004010FC
00401100
00401104
00401108
0040110C
00401110
00401114
00401118
0040111C
00401120
00401124
00401128
0040112C
00401130
00401134
00401138
0040113C
00401140
00401144
00401148
0040114C
00401150
00401154
00401158
0040115C
00401160
00401164
00401168
0040116C
00401170
00401174
00401178
0040117C
00401180
00401184
00401188
0040118C
00401190
00401194
00401198
0040119C
004011A0
004011A4
004011A8
004011AC
004011B0
004011B4
004011B8
004011BC
004011C0
004011C4
004011C8
004011CC
004011D0
004011D4
004011D8
004011DC
004011E0
004011E4
004011E8
004011EC
004011F0
004011F4
004011F8
004011FC
00401200
00401204
00401208
0040120C
00401210
00401214
00401218
0040121C
00401220
00401224
00401228
0040122C
00401230
00401234
00401238
0040123C
00401240
00401244
00401248
0040124C
00401250
00401254
00401258
0040125C
00401260
00401264
00401268
0040126C
00401270
00401274
00401278
0040127C
00401280
00401284
00401288
0040128C
00401290
00401294
00401298
0040129C
004012A0
004012A4
004012A8
004012AC
004012B0
004012B4
004012B8
004012BC
004012C0
004012C4
004012C8
004012CC
004012D0
004012D4
004012D8
004012DC
004012E0
004012E4
004012E8
004012EC
004012F0
004012F4
004012F8
004012FC
00401300
00401304
00401308
0040130C
00401310
00401314
00401318
0040131C
00401320
00401324
00401328
0040132C
00401330
00401334
00401338
0040133C
00401340
00401344
00401348
0040134C
00401350
00401354
00401358
0040135C
00401360
00401364
00401368
0040136C
00401370
00401374
00401378
0040137C
00401380
00401384
00401388
0040138C
00401390
00401394
00401398
0040139C
004013A0
004013A4
004013A8
004013AC
004013B0
004013B4
004013B8
004013BC
004013C0
004013C4
004013C8
004013CC
004013D0
004013D4
004013D8
004013DC
004013E0
004013E4
004013E8
004013EC
004013F0
004013F4
004013F8
004013FC
00401400
00401404
00401408
0040140C
00401410
00401414
00401418
0040141C
00401420
00401424
00401428
0040142C
00401430
00401434
00401438
0040143C
00401440
00401444
00401448
0040144C
00401450
00401454
00401458
0040145C
00401460
00401464
00401468
0040146C
00401470
00401474
00401478
0040147C
00401480
00401484
00401488
0040148C
00401490
00401494
00401498
0040149C
004014A0
004014A4
004014A8
004014AC
004014B0
004014B4
004014B8
004014BC
004014C0
004014C4
004014C8
004014CC
004014D0
004014D4
004014D8
004014DC
004014E0
004014E4
004014E8
004014EC
004014F0

Normalized representation in computer

$$r_{10} = \pm M * 2^E, \text{ where } 1 \leq M < 2$$

$$M = (1.b_1b_2b_3...b_n)_2$$

M: significant, E: exponent



$$(1.b_1b_2b_3...b_n)_2$$

Pointer in C

- A pointer is just a variable with an unique interpretation.
 - The memory address(ADDR) of that variable.
 - The number of bytes used by the variable.
 - How to interpret the content stored in ADDR.
 - It represents a memory address.

Pointer in C

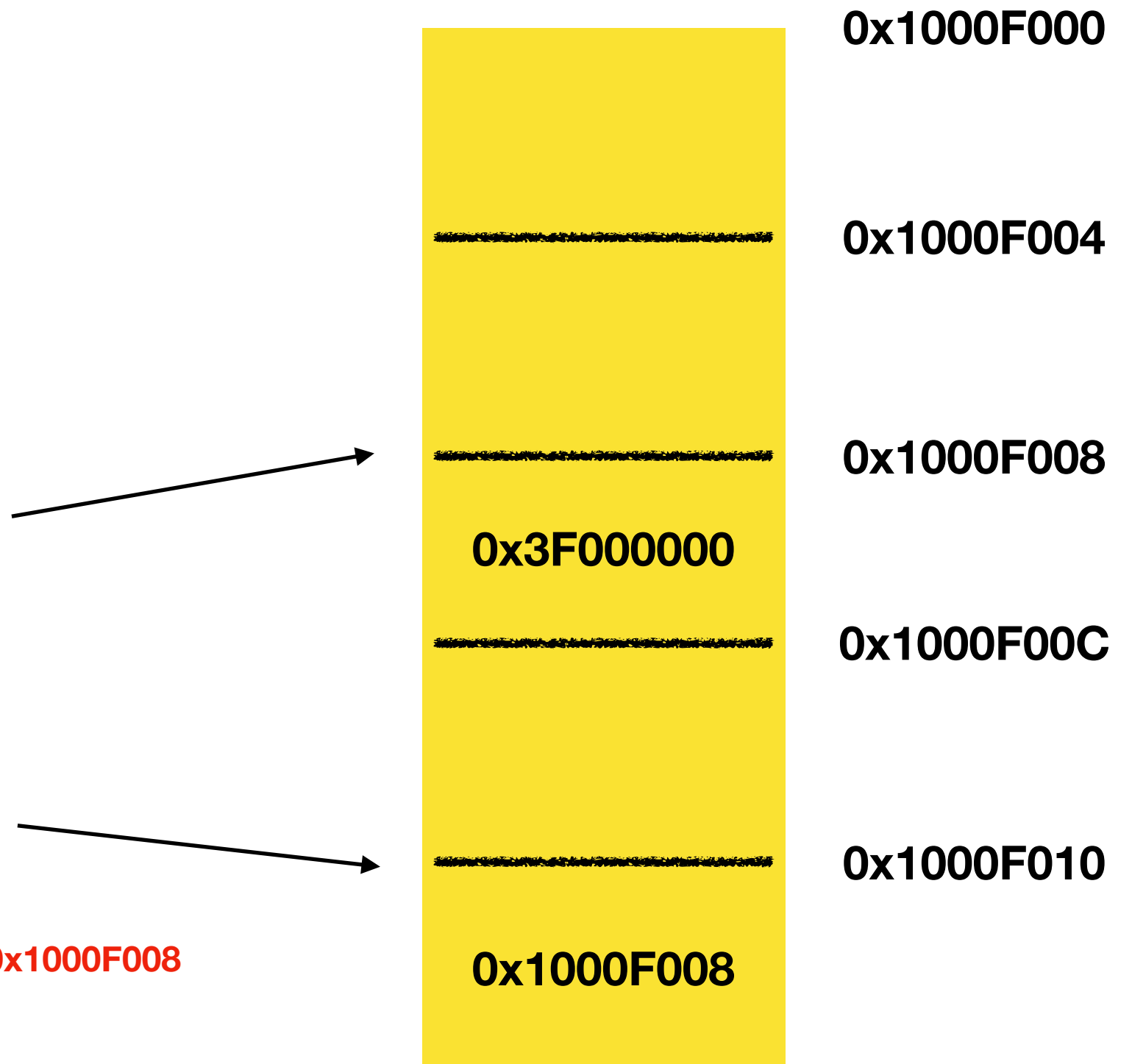
```
unsigned int a = 0x3F000000;
```

```
*(0x1000F008) = 0x3F000000
```

```
unsigned int *ptr = &a;
```

```
printf("%p", ptr);   *(0x1000F010) = 0x1000F008
```

```
printf("%u", *ptr);   *(*(0x1000F010)) = *(0x1000F008) = 0x3F000000
```



Array in C

- An array is just a variable with different byte lengths.
 - The memory address(ADDR) of that variable.
 - The number of bytes used by the variable.
 - ▶ It contains multiple values.
 - How to interpret the content stored in ADDR.
 - ▶ Compiler doesn't deference it unless you use [] symbol.

Pointers in C

```
unsigned int grades[] = {0, 1, 2, 4, 8};
```

```
unsigned int *ptr = grades;
```

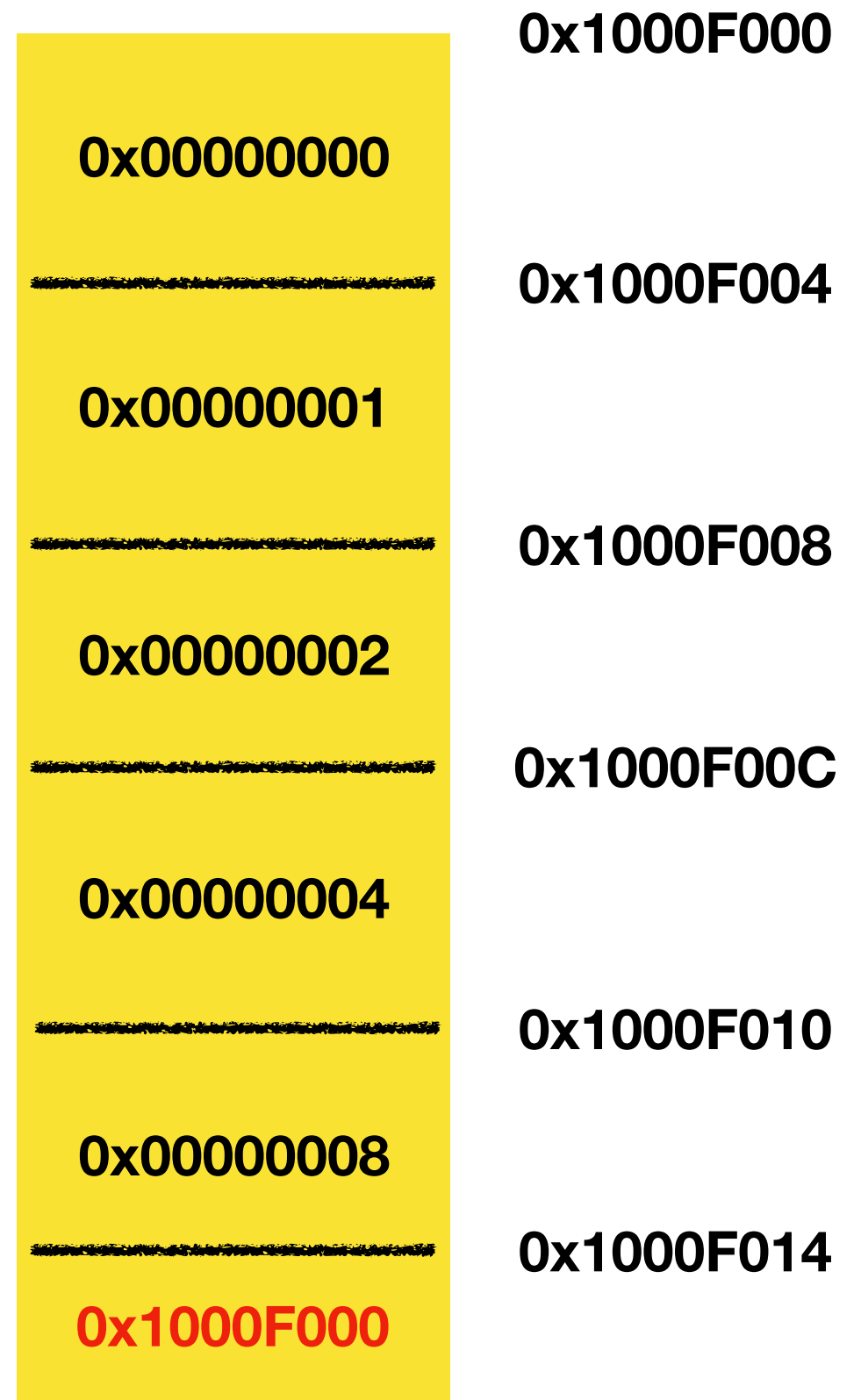
**(0x1000F004) = 0x1000F000*

```
grades = ptr;
```


```
grades[2]
```

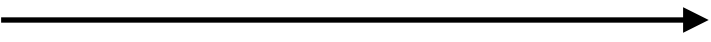
```
*(ptr + 2)
```



**(0x1000F000 + 8) = 0x2*



Exercise 1 (How to print 15)

```
void func1(unsigned num) {  
    num += 1;        num(different num from main) += 1  
}
```

```
void func2(unsigned* num) {  
    *num += 1;        *(0x7F100000) += 1  
}
```

```
int main() {  
    unsigned num = ?;  
func1(num);  func1(?);  
    func2(&num);  func2(0x7F100000);  
    printf("%u", num);  
}
```

num + 1 = 15

Lab1 : part1, par2.c, part4.c, part5.c

Exercise 2 (How to print 8)

```
int main() {  
    unsigned short array[] = {0, 2, 4, 8, 16};  
    int i = ?;  
    int j = ?;  
    unsigned short *ptr1 = &(array[i]);  
    unsigned short *ptr2 = &(array[j]);  
    unsigned int ptr1_int = (unsigned int) ptr1;  
    unsigned int ptr2_int = (unsigned int) ptr2;  
    printf("%u\n", ptr2_int - ptr1_int);  
}
```

ptr2_int - ptr1_int == 8

(the address stored in ptr2) - (the address stored in ptr1) == 8

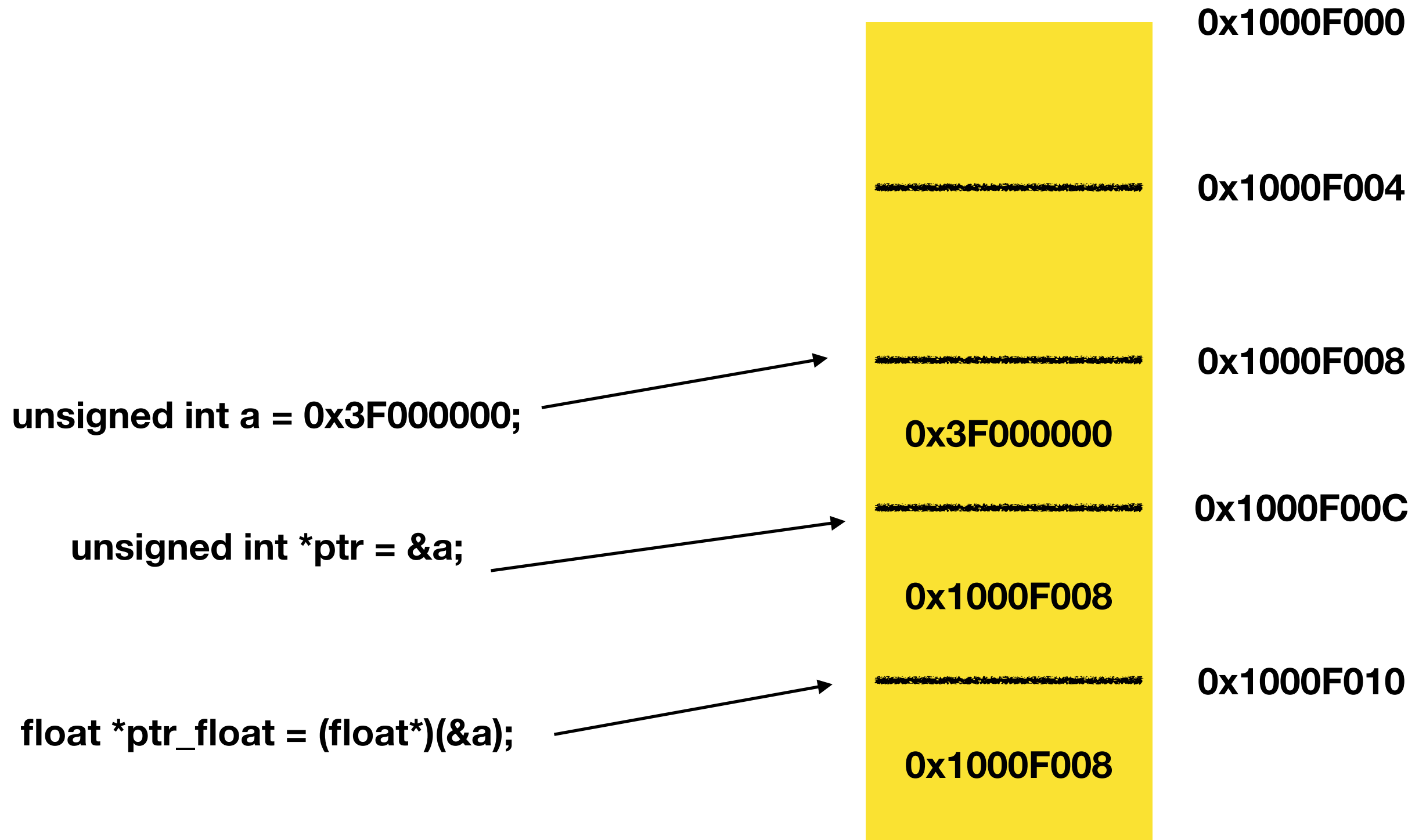
(the address of array[j]) - (the address of array[i]) == 8

unsigned short is two bytes

j - i == 4

Lab1 : part1, par2.c, part3.c, part4.c

Pointers in C



Exercise 3 (How to print 1056964608)

```
int main() {  
    float answer = ?;  
  
    unsigned* unsigned_ptr = (unsigned*)&answer;  
    printf("%u\n", *unsigned_ptr);  
}
```

1056964608 = 0x3F000000

What is 0x3F000000 when interpreting it as a float?

0.5

Exercise 4 (How to print 262144)

```
int main() {  
    unsigned answer = ?  
  
    answer = (answer >> 3) << 5;  
    printf("%u\n", answer);  
}
```

262144 = 0x00040000 = b 0000,0000,0000,0100,0000,0000,0000,0000

do >> 5 : b ????,?000,0000,0000, **0010**,0000,0000,0000

do << 3 : b ??00,0000,0000,**0001**, 0000,0000,0000, 0???

One more thing to know for lab 1: 0x00FF1112 & 0xFF == 0x00000012

Lab1 : part6.c

Lab1

- The first part, “mini”, ask you to implement some functions to let you be familiar with basic C language syntax, including pointer.
- The second part, “scratch”, ask you to implement three programs from scratch.
 - You need to write a Makefile for each program.
 - Each program must contain a main() function (why?).
 - ▶ You also need to understand to how to use the arguments of main().
 - You need to learn C standard file I/O library.

Lab1

- You may need to learn `qsort()` of C standard library.
 - Or implement a hash table library by yourself.
- How to learn these libraries?
 - Read “The C Programming Language”.
 - Use `man`, e.g., `man fopen`.
 - Google.