OS n' SP extension with respect to C pointer

Halloween's Episode— No treats, Only tricks

E

University of Birmingham

2023 Winter

Table of Contents

Table of Contents

What address is?

- There is 2^m *bytes* of memory, so m bits can represent the whole memory space.

What address is?

- There is 2^m *bytes* of memory, so m bits can represent the whole memory space.
- Every resident/cell is one byte.

What address is?

- There is 2^m *bytes* of memory, so m bits can represent the whole memory space.
- Every resident/cell is one byte.
- Modern 8 bytes of size for pointer—8*8=64 bits.
  $2^64 = 16EiB$

What address is?

- There is 2^m *bytes* of memory, so m bits can represent the whole memory space.
- Every resident/cell is one byte.
- Modern 8 bytes of size for pointer—8*8=64 bits.
  $2^64 = 16EiB$
- "8 byte" is so intrinsic that it has a special name, doubleword.
  4 bytes — word

What array is

A consecutive memory space. The name is synonymous to the address of the first element of the array.

A concept: L-value

- int a; // a has address &a
  int *p;// p has address &p;

A concept: L-value

- int a; // a has address &a
  int *p;// p has address &p;
- a and p, we name as L-value:
  A L-value is an object that occupies some identifiable location in memory (Legal operand of &).

A concept: L-value

- int a; // a has address &a
  int *p;// p has address &p;
- a and p, we name as L-value:
  A L-value is an object that occupies some identifiable location in memory (Legal operand of &).
- int arr[4];//Is arr a L-value?

A concept: L-value

- int a; // a has address &a
  int *p;// p has address &p;
- a and p, we name as L-value:
  A L-value is an object that occupies some identifiable location in memory (Legal operand of &).
- int arr[4];//Is arr a L-value?
- &arr; //This operation is definitely LEGAL

What array is

- arr[0]==*arr; &arr[0]==a;
  What is &arr? Try!

What array is

- arr[0]==*arr; &arr[0]==a;
  What is &arr? Try!
- Assignment to arr is illegal, unlike ordinary variables.

What array is

- arr[0]==*arr; &arr[0]==a;
  What is &arr? Try!
- Assignment to arr is illegal, unlike ordinary variables.
- error: assignment to expression with **array type**

What array is

- arr[0]==*arr; &arr[0]==a;
  What is &arr? Try!
- Assignment to arr is illegal, unlike ordinary variables.
- error: assignment to expression with **array type**
- Meanwhile, it returns address(p=array;), but its address is again the same address.

What array is

- arr[0]==*arr; &arr[0]==a;
  What is &arr? Try!
- Assignment to arr is illegal, unlike ordinary variables.
- error: assignment to expression with **array type**
- Meanwhile, it returns address(p=array;), but its address is again the same address.
- Try some arithmetic, like +1, comparing &arr+1 and arr+1.

What array is from compiler's point of view

"array" is not a pointer but reduced to pointer:

> *Except when it is the operand of the sizeof operator, the _Alignof operator, or the unary & operator, or is a string literal used to initialize an array, an expression that has type "array of type" is converted to an expression with type "pointer to type" that points to the initial element of the array object and is* **not an lvalue**. *If the array object has register storage class, the behavior is undefined.*
> *—C11 6.3.2.1*

What pointer is?

A variable which has an address as value. n bits in n system, and n is elementary unit on which CPU do computation.

Pointer and function: As parameters and references

- C can only pass by value

Pointer and function: As parameters and references

- C can only pass by value
- It is why scanf needs pointer as parameter

Pointer and function: As parameters and references

- C can only pass by value
- It is why scanf needs pointer as parameter
- We can also have pointer to function

Pointer and function: As parameters and references

- C can only pass by value
- It is why scanf needs pointer as parameter
- We can also have pointer to function
- Then pointer array to functions

Pointer and function: As parameters and references

- C can only pass by value
- It is why scanf needs pointer as parameter
- We can also have pointer to function
- Then pointer array to functions
- Then pointer to function as parameter (for a general function)

Pointer Or Array as formal parameter

fun(char a[]) and fun (char *a) is identical, not just synonymous.
Try some arithmetic on a.

Table of Contents

Pointer Or Array as formal parameter

- int a=5;
  int b[3];
  int c=6;

Pointer Or Array as formal parameter

- int a=5;
  int b[3];
  int c=6;
- b[-1]=0;
  Try

Pointer Or Array as formal parameter

- int a=5;
  int b[3];
  int c=6;
- b[-1]=0;
  Try
- It is your duty to ensure legal access, not compiler's

Pointer Or Array as formal parameter

- int a=5;
  int b[3];
  int c=6;
- b[-1]=0;
  Try
- It is your duty to ensure legal access, not compiler's
- But runtime concerns, try b[3]=3

Table of Contents

Useful tricks

- Simulating 2-dimensional array
  #define A(r, c) (A[(r)*WIDTH + (c)])
- flexible array member—struct hack