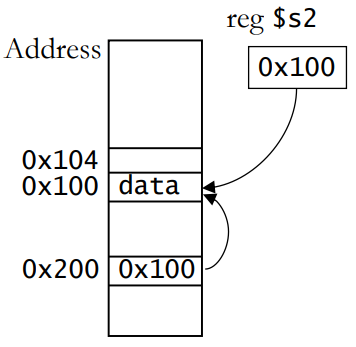
Computer Systems Lecture 8

Pointers

We have seen pointers in assembly: **lw $t1, 0($s2**).$s2 points to the location in memory where the actual data is kept. $s2 is a register but there nothing stopping us from having pointers stored in memory like normal variables.

C Pointers

A C pointer is a variable that holds the address of a piece of data.

Pointers are declared using **\***:

**int \*p;**  creates a pointer to an integer, the compiler needs to know what data type the pointer points to.

Basic pointer usage:

**p = &I; // p now points to the location of I in memory**

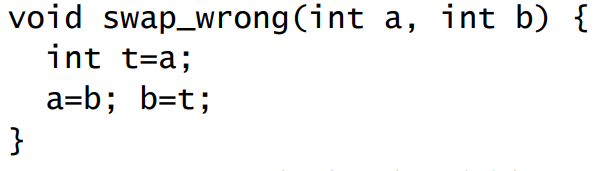
**\*p = 5; // This takes p (now pointing to i) and tells it to access the memory and sets it to 5**

Pointers as Function Arguments

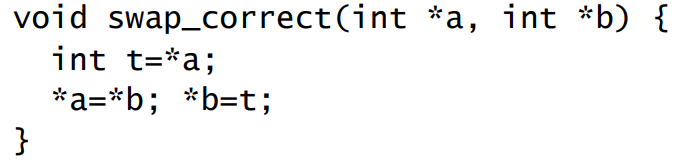
In Java, an argument with primitive type is passed by value and an argument with class type is passed by reference.

In C, all arguments are passed by value, to get the effect of passing by reference we pass a pointer as an argument.

The Swap Function



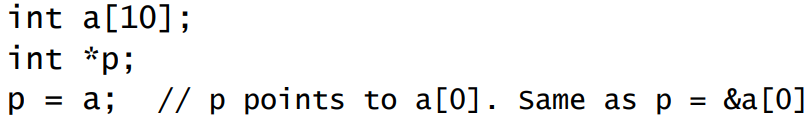
Swap\_wrong swaps the local variables a and b which are unknown outside of the function



This is the way swap should be implemented, save the value at a in t, set the value of a to the value of b, then set the value of b to the value in t.

Pointer Arithmetic and Arrays

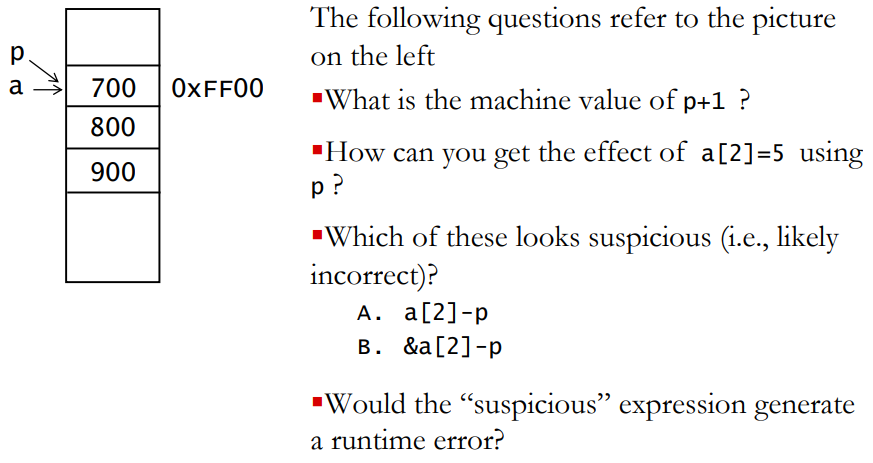
C allows arithmetic on pointers:



P+1 points to a[1] ( &a[1] = &a[0]+1) ( the compiler multiplies +1 with the data types size needed).

In general p+I points to a[i] and \*(p+i) is a[i]

You can also simply use the array (\*(a+i) is the same as a[i] as is p[i]) (the difference between a and p is that you can’t change what a points to but you can p).

* p+1 = 0xFF00 + word size
* \*(p+2) = 5
* B but A is the one that would break
* no