

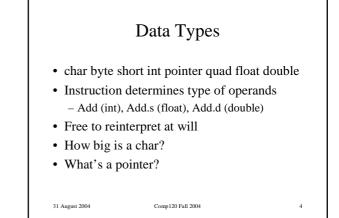


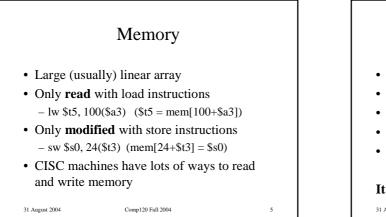
- Program and data have the same representation
- Programs can manipulate programs
- Programs can manipulate themselves!

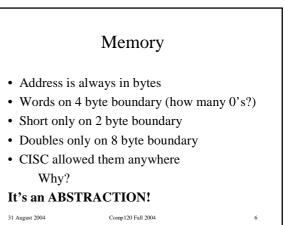
Comp120 Fall 2004

• Bits not the only way (Lisp)

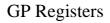
31 August 2004







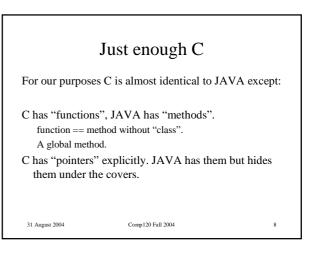
1

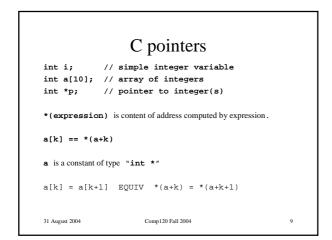


- Variables for our programs
- The ONLY operands for most instructions
- A very small number (32 in MIPS) Why?
- All 32 bits
- What about new 64 bit ISA's?

31 August 2004

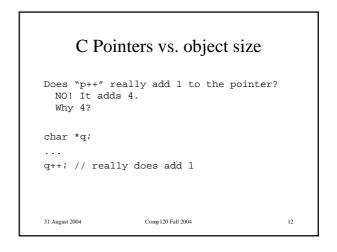
Comp120 Fall 2004

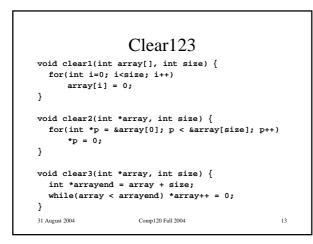


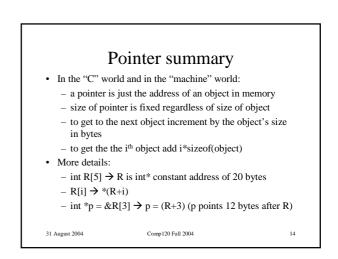


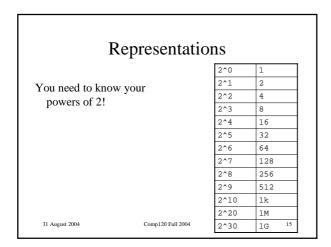
Legal uses of C Pointers			
int i;	// simple integer variable		
int a[10];	<pre>// array of integers</pre>		
int *p;	<pre>// pointer to integer(s)</pre>		
p = &i	// & means address of		
p = a;	// no need for & on a		
p = &a[5];	// address of 6^{th} element of a		
*p	// value of location pointed by p		
*p = 1;	<pre>// change value of that location</pre>		
*(p+1) = 1;	<pre>// change value of next location</pre>		
p[1] = 1;	<pre>// exactly the same as above</pre>		
p++;	// step pointer to the next element		
31 August 2004	Comp120 Fall 2004	10	

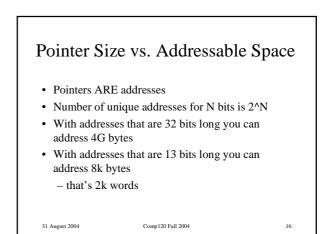
Legal uses of Pointers	5		
<pre>int i; // simple integer variab int a[10];// array of integers int *p; // pointer to integer(s)</pre>			
So what happens when p = &i What is value of p[0]? What is value of p[1]?			
31 August 2004 Comp120 Fall 2004	11		

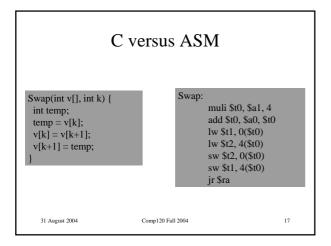


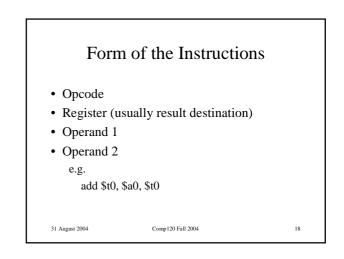


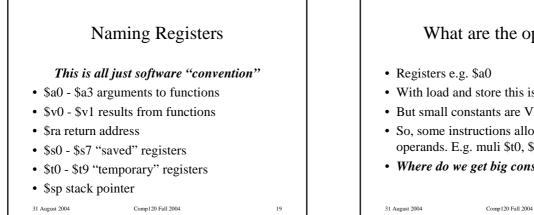












What are the operands?

- With load and store this is logical enough
- But small constants are VERY common
- So, some instructions allow "immediate" operands. E.g. muli \$t0, \$a1, 4
- Where do we get big constants?

20

