How Does Union/Intersect

setA ???????? ???????? ???????? 01101011 01010100

? 9 ? 6 ?
8 2 13
14 ? 4
11

setB 01100011 01010100

setA.union (setB)

9 6
8 2
14 13
4

setA.intersect (setB)

Defining the Operators

- The union, intersection and difference operations act on 2 sets to produce a third set. What's tricky is making sure these methods work correctly when they operate on sets of different sizes. For example,

```
Bitset union (Bitset setB)
{   Bitset temp = new Bitset (maxSize  > setB.maxSize ? this : setB);

    int nbyte = Math.min (byteArray.length, setB.byteArray.length);

    for (int i = 0; i < nbyte; i++)
    {
        temp.byteArray[i] = (byte)  (byteArray[i] | setB.byteArray[i]);
    }
    return temp;
}
```


Other Useful Set

- Frequently we need to operate on a single element of a set: to add it to the set, remove it from the set, or test whether it is present in the set. All these things are easy to do in terms of primitive **Bitset** operations:

```
boolean member  (int i)
{
    if (i >= maxSize)  return false;
    else return (getBit (i) );
}
```

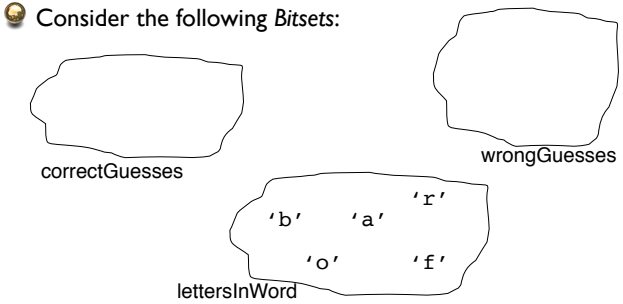
```
void  include (int i)
  {  if  (i >= maxSize) error("Too big!");  setBit(i);  }
```

```
void exclude (int i)
  {  if (i  >= maxSize) error ("Too big!");  clearBit(i);  }
```

- All the above check parameter i's validity.

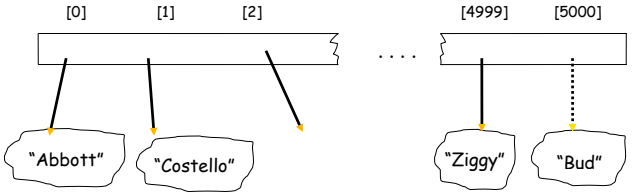
Hangman: A Bitset Application

- Consider the following *Bitsets*:

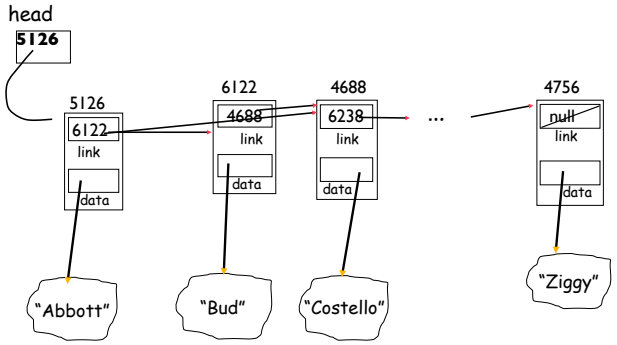


Limitations of Arrays, *revisited*

- Consider employee objects, sorted alphabetically by last name.
  - How do we add / delete objects from the array?

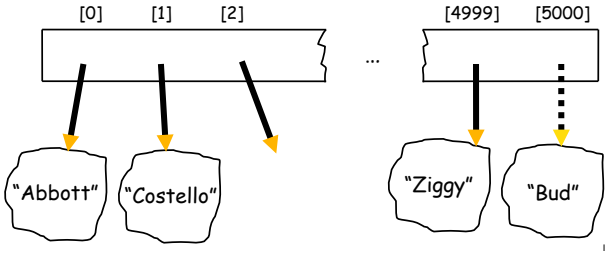


Linked-List Alternative

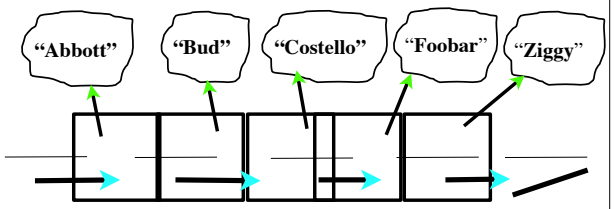


Limitations of Arrays, revisited

- Consider "Employee" objects, sorted alphabetically by last name.
  - How do we add / delete objects from the array?



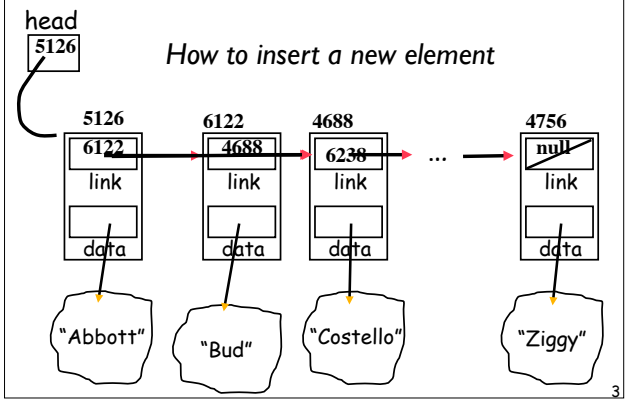
Linked-List as an Alternative




```
public class ListNode
{
     String  data;     // the green arrows
     ListNode link;    // the blue arrows
}
```

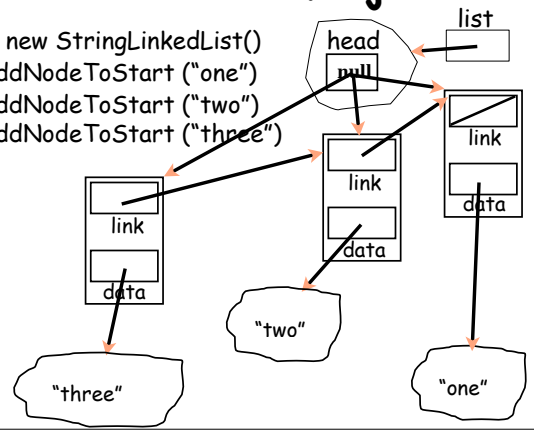

Linked-Lists, continued

How to insert a new element

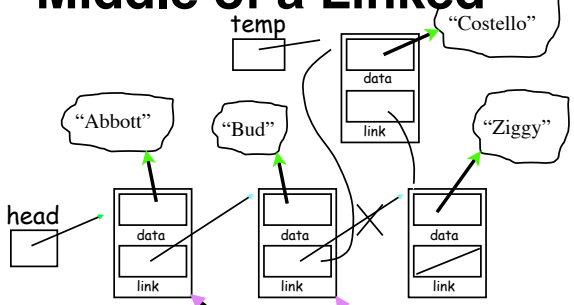


LinkedListDemo.java

list = new StringLinkedList()
list.addNodeToStart ("one")
list.addNodeToStart ("two")
list.addNodeToStart ("three")



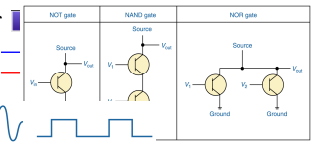
Inserting in the Middle of a Linked-



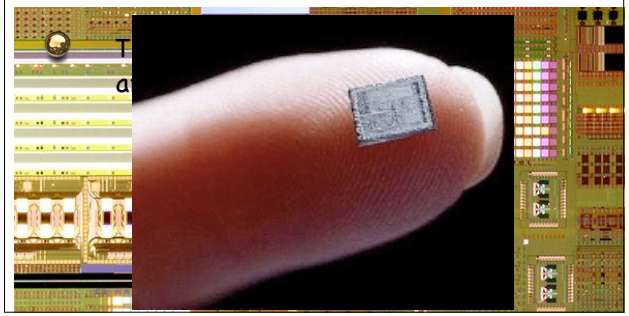
pp.setLink(temp);
temp.setLink(p);

Managing Complexity

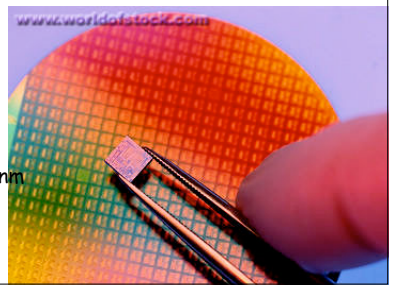
- ABSTRACTION: Simplifying assumptions that ignore or gloss over details that could be articulated in a more detailed analysis.
- Examples from computing
  - user-level commands Cancel Save
  - higher-level language y = *(px) ++; ...
  - machine language load x, R3 ...
  - register-level behavior
  - clocked digital logic
  - semiconductors
  - analog voltages



Computer Hardware

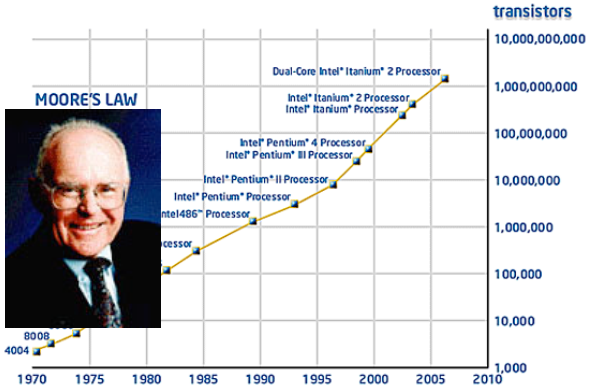


Silicon Wafers and Chips

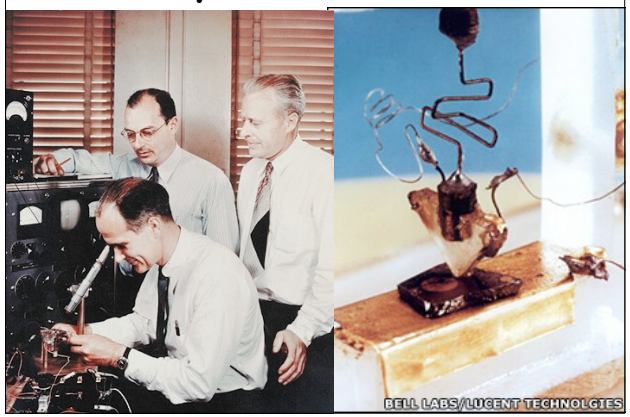


A nanometer is one billionth of a meter
A human hair = 90,000nm
Ragweed pollen = 20,000nm
Bacteria = 2,000nm

Exponential Growth



History of the Transistor



Machine Architecture

- The MIPS Processor Family
  - A RISC architecture embodied by the R2000, R3000, R4000 and R6000 processors.
  - MIPS Technologies = principal architect of embedded 32- and 64-bit RISC processors, licensed to system OEMs, semiconductor manufacturers, etc.
  - See Britton: "Mips Assembly Language Programming" and **http://www.cs.wisc.edu/~larus/spim.html**
- Processor Performance
  - *Time per Task* = C * T * I
    - C = **Cycles per instruction**
    - T = **Time per cycle (clock speed)**
    - I = **Instructions per Task**

Number Systems

- **Unsigned Binary Numbers**

```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 0
0 0 0 0 0 0 1 1
0 0 0 0 0 1 0 0
        ...
        ...
1 1 1 1 1 1 1 1
```

- **Bit Numbering**

0	0	0	0	0	1	0	0
b7	b6	b5	b4	b3	b2	b1	b0

Binary Arithmetic

- The usual algorithm works, with 1 + 1 = 0 (carry 1):

```
          1 1
  0 0 0 0 0 1 0 1
+ 0 0 0 0 1 1 1 0
  ---------------
  0 0 0 1 0 0 1 1
```

- Carry out of the MSB means the sum is too large to represent.
- Conversion of binary to decimal involves adding up the decimal values of the powers of 2 corresponding to the 1 bits. For example,
  - 0 1 0 1 1 0 1 0

Signed Integers

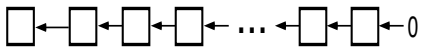
- How many different integers can be represented using N bits?
- To "negate" an integer: form the two's-complement by
  - Taking the logical complement
  - Adding one to the logical complement
  - For example,

```
 0 0 0 0 0 1 1 0
 1 1 1 1 1 0 0 1
+               1
---------------
 1 1 1 1 1 0 1 0
```

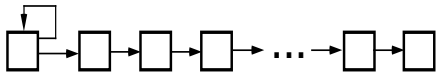
- What about negating zero?
- What is 1 1 1 1 1 1 1 1 ?
- What is 1 0 0 0 0 0 0 0 ?

Doubling / Halving By Shifting

- *Doubling* is accomplished via "arithmetic left shift"
  - Slide all bits one place to the LEFT, and make the new b_0 equal to 0.



- *Halving* is accomplished via "arithmetic right shift"
  - Slide all bits one place to the RIGHT, and make the new b_{n-1} stay the same.



Hexadecimal (base 16)

- The 16 patterns of 4 bits are as follows:

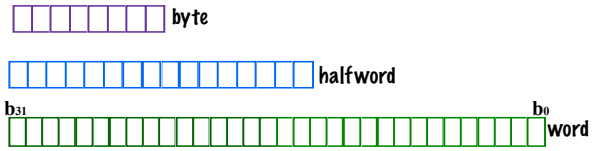
Binary	0000	0001	0010	0011	0100	0101	0110	0111
Decimal	0	1	2	3	4	5	6	7
Hex	0	1	2	3	4	5	6	7

Binary	1000	1001	1010	1011	1100	1101	1110	1111
Decimal	8	9	10	11	12	13	14	15
Hex	8	9	A	B	C	D	E	F

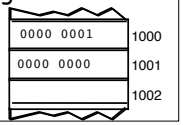
- So an 8-bit -1 would be FF. To distinguish such numbers from identifiers, we precede them by "0x" in C/C++ and in Java.
  - `0x69 =`

Data Sizes

- The smallest unit of data that can be stored in / retrieved from memory is 1 byte = 8 bits

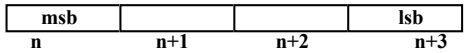


- Memory addresses are byte addresses; ask for the byte at location 1001, and you'll get it.
  - *Word* addresses must be divisible by 4
  - *Halfword* addresses must be divisible by 2

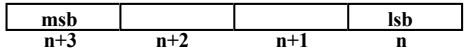


The "Endian" Wars

- Big Endian: Most significant byte is in lower-numbered address
  - e.g., M680x0, IBM mainframes, PowerPC, SPARC



- Little Endian: Least significant byte is in lower-numbered address
  - e.g., Intel 80x86 and Core Duo, DEC Alpha & VAX



- Total incompatibility between the two! BUT …
  - MIPS is bi-endian!