Recitation0

Outline

- About me
- Autolab
- Fish Machines
- ssh
- Writing code
- Datalab

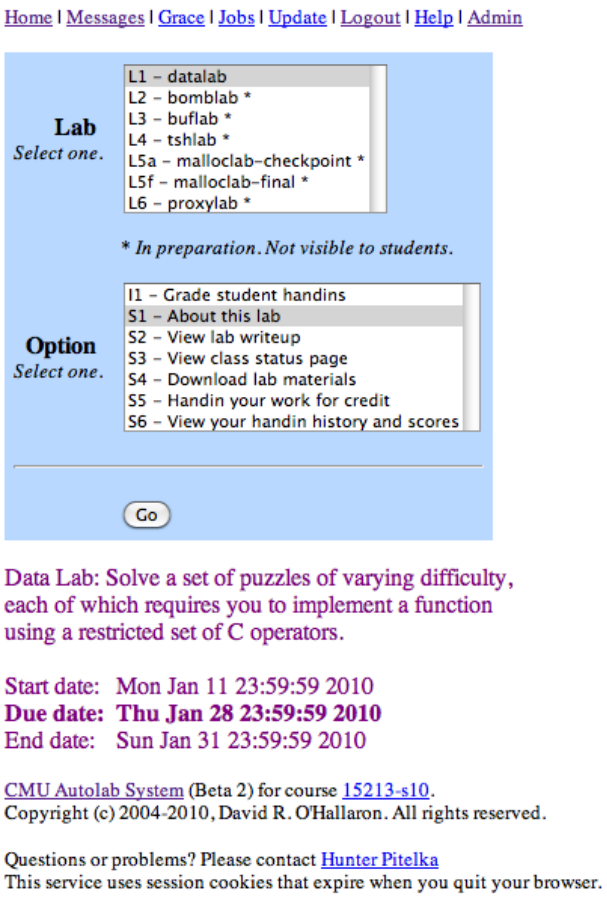
About me

- ECE IMB 5th year
- Last Semester!
- jprimero@andrew.cmu.edu
- Office Hours
  - 6-9 Thursday in Wean 5207

autolab

- Serves as a portal for:
  - Lab Materials
  - Grading
  - Forums
  - Class Status
  - Friendly Competition

- http://autolab.cs.cmu.edu

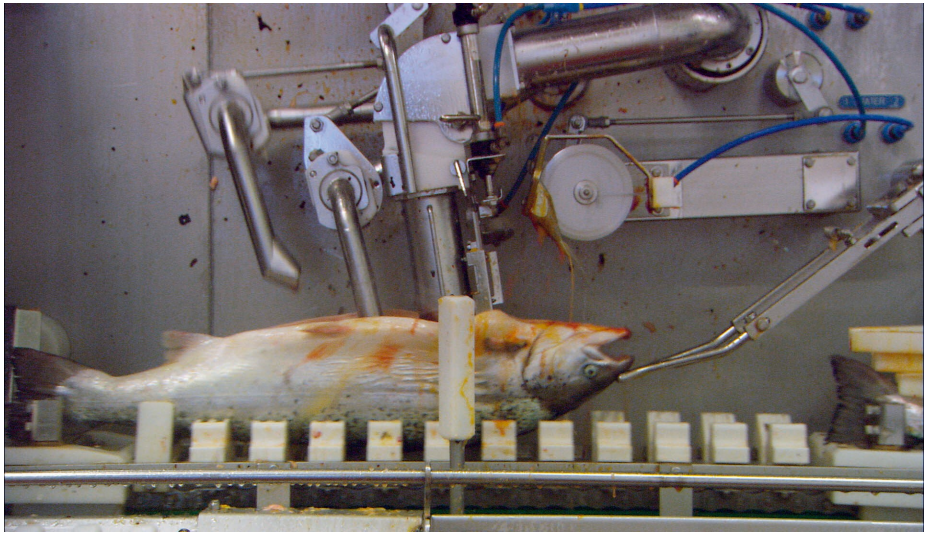


Autolab TODO

- TEST YOUR AUTOLAB ACCOUNT
- If your account is not working, send an email to the staff
  - 15-213-staff@cs.cmu.edu
- Datalab is available now!

Fish machines

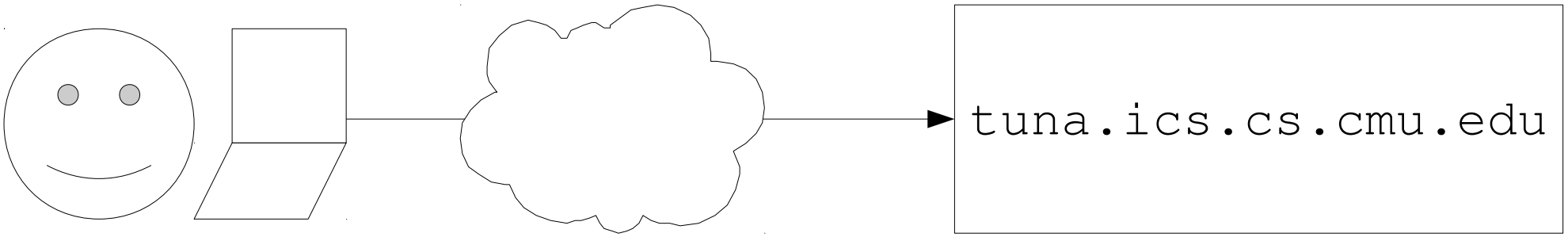
- Powerful computer cluster donated by Intel
- Your labs will be graded on the fish machines
- Must “ssh” into machines



Fish machine!

ssh

- Allows one to login to a machine remotely and control it

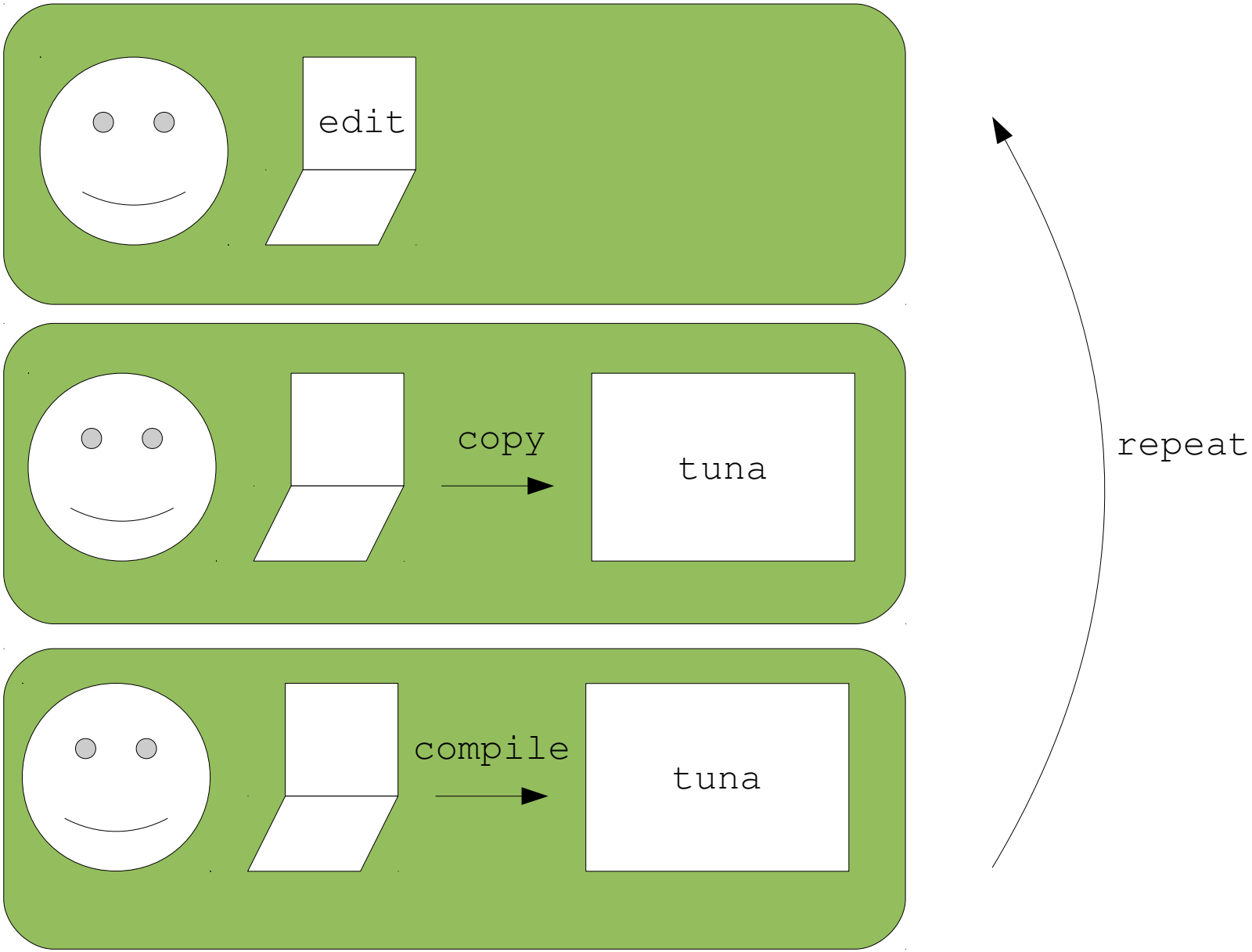



```
$ssh -x -l jprimero@ANDREW.CMU.EDU tuna.ics.cs.cmu.edu
```

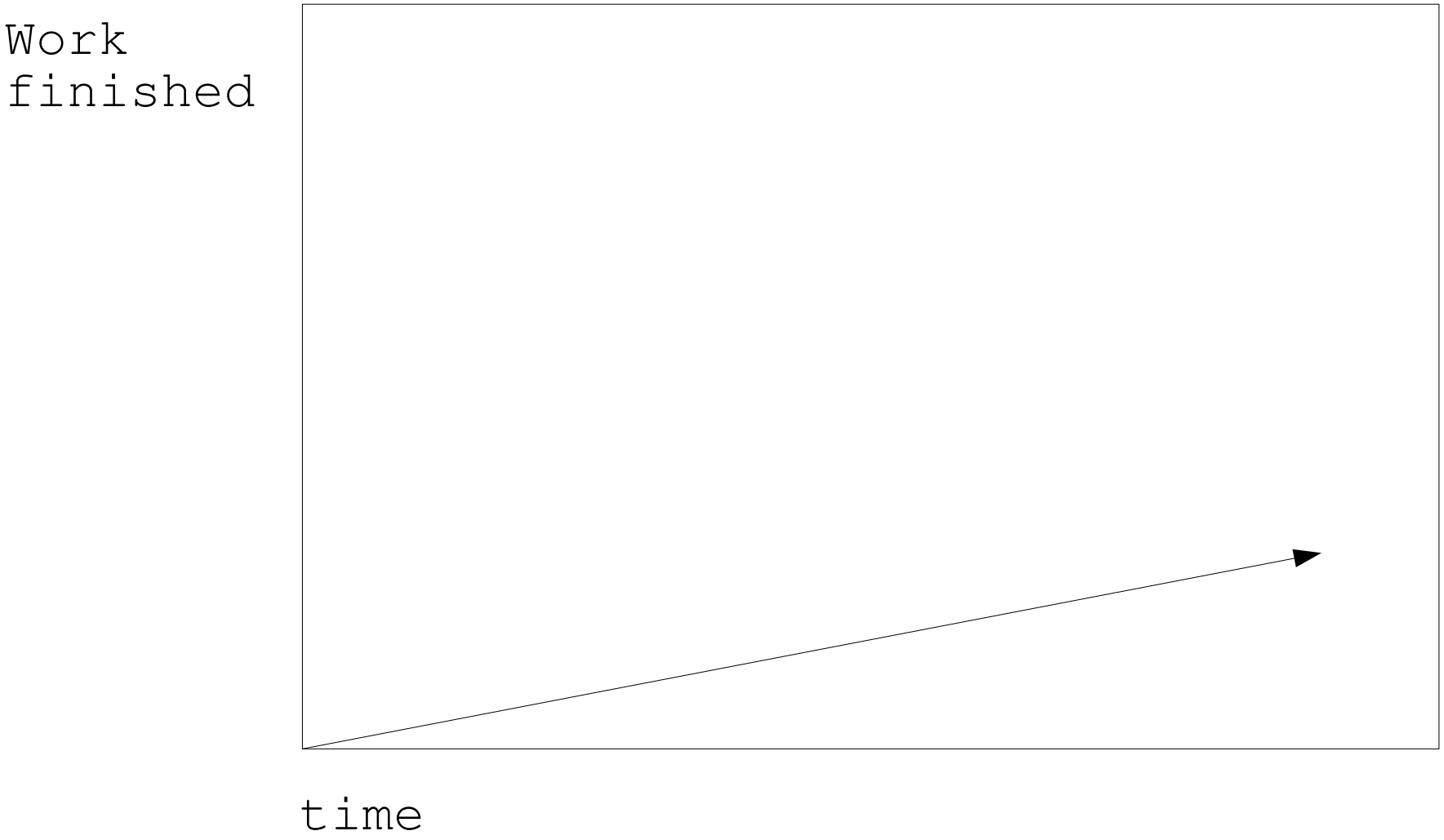
ssh clients

- Windows
  - Putty
  - SSHClient
  - Cygwin
- Mac/Linux
  - Just `$ssh -x -l jprimero@ANDREW.CMU.EDU tuna.ics.cs.cmu.edu`

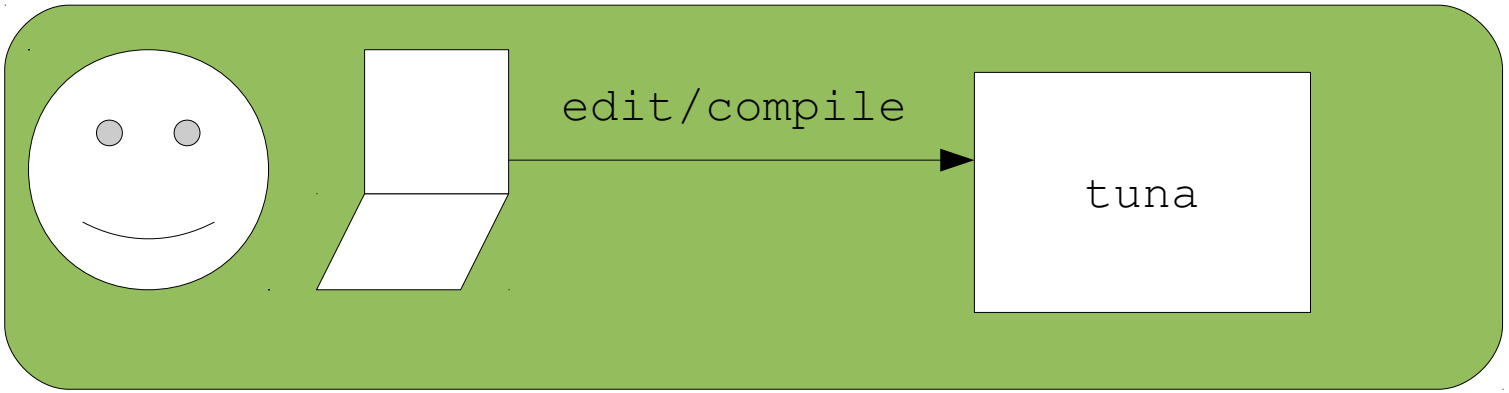
A workflow I see all too often



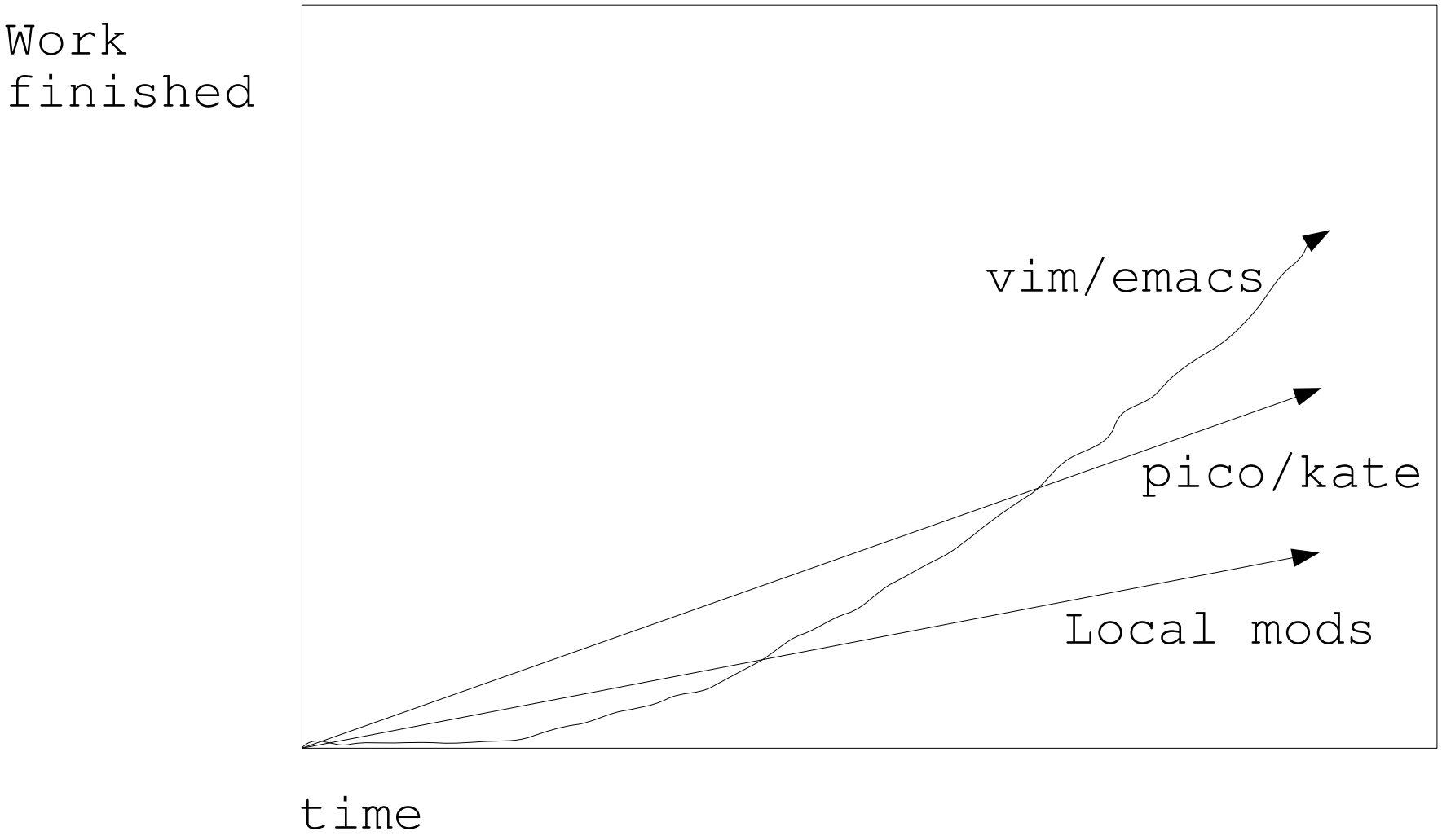
Efficiency Graph



Much better workflow



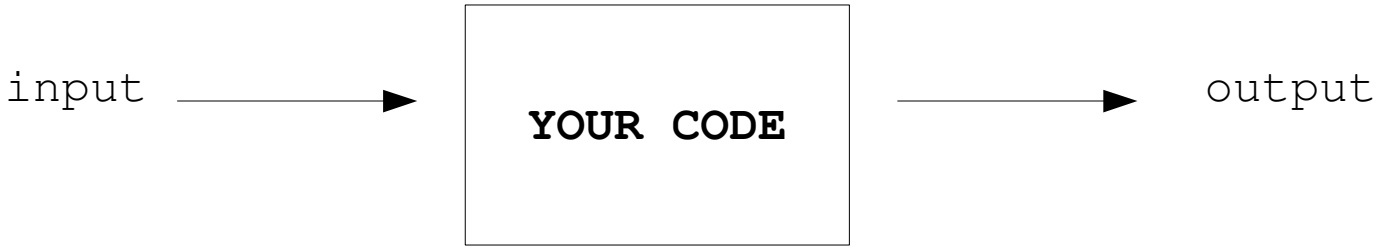
Efficiency Graph



Note: This is all based on my humble opinion
so do what you want with this info

datalab

- A set of 13 fun puzzles!
- Each puzzle requires you to return an output based on some input



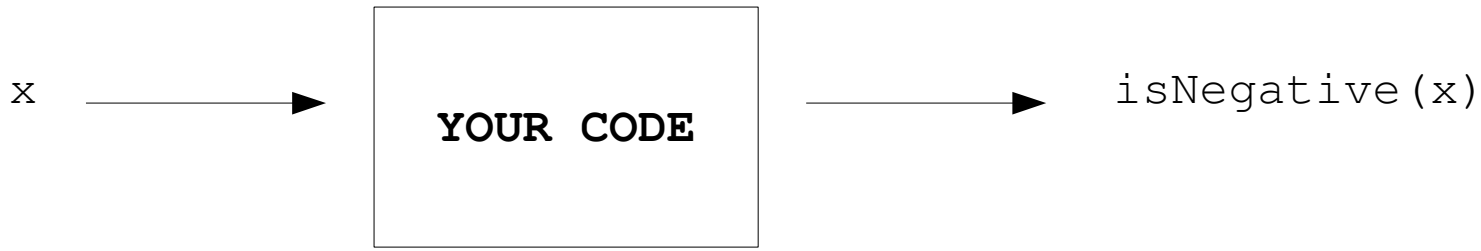
Datalab rules

- You can only use a set of bitwise operators for each problem
- Only straight-line code!
  - No 'if', 'for', 'while'
  - Kinda tricky!

TA

Datalab example

```
/*
 * isNegative - returns 1 if x is negative
 * Examples: isNegative(5) = 0, isNegative(-7) = 1
 * Legal ops: ! ~ & ^ | + << >>
 * Max ops: 5
 * Rating: 1
 */
int isNegative(int x) {
  return 2;
}
```

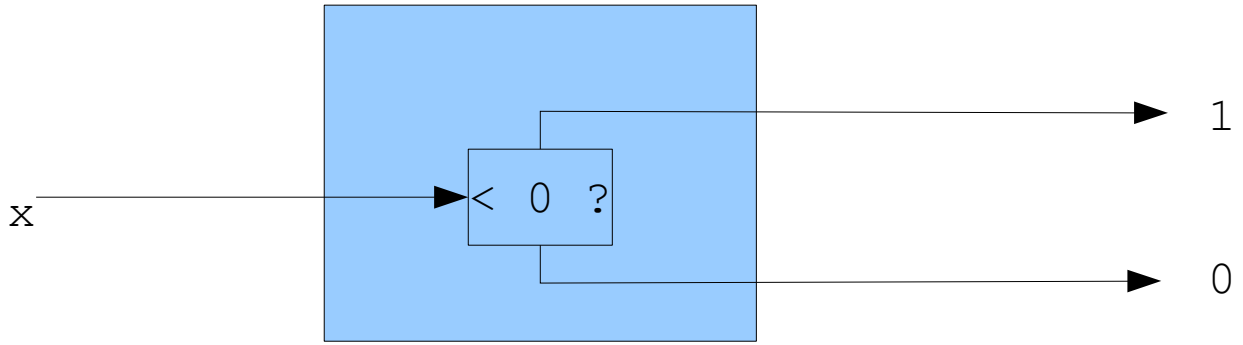


Easy solution

```
/*
 * isNegative - returns 1 if x is negative
 * Examples: isNegative(5) = 0, isNegative(-7) = 1
 * Legal ops: ! ~ & ^ | + << >>
 * Max ops: 5
 * Rating: 1
 */
int isNegative(int x) {
  if (x < 0) {
    return 1;
  } else {
    return 0;
  }
}
```

Branching

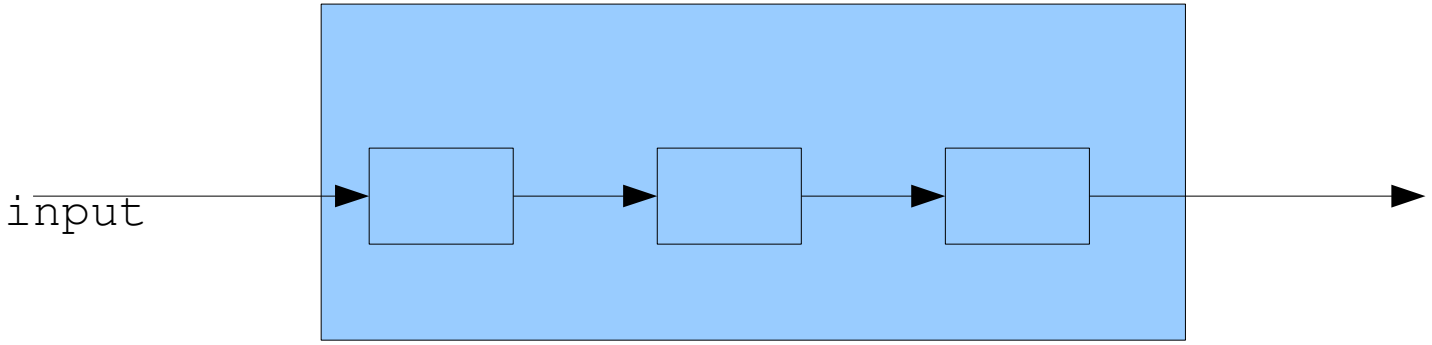
```
int isNegative(int x) {
  if (x < 0) {
    return 1;
  } else {
    return 0;
  }
}
```



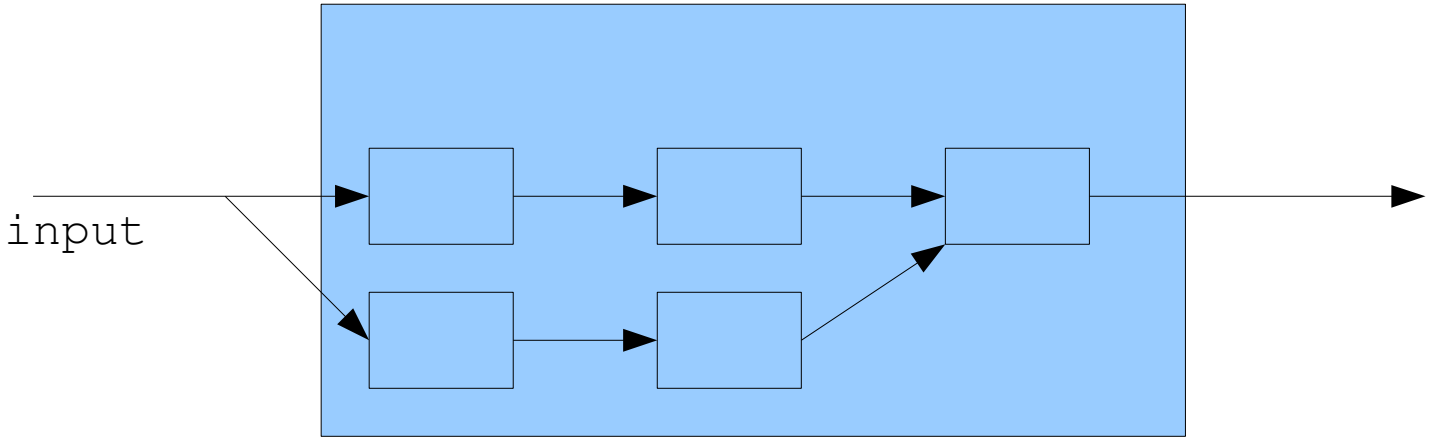
Want Straight-line code!

Straight line code

- For you ECE majors, think of straight line code as a circuit



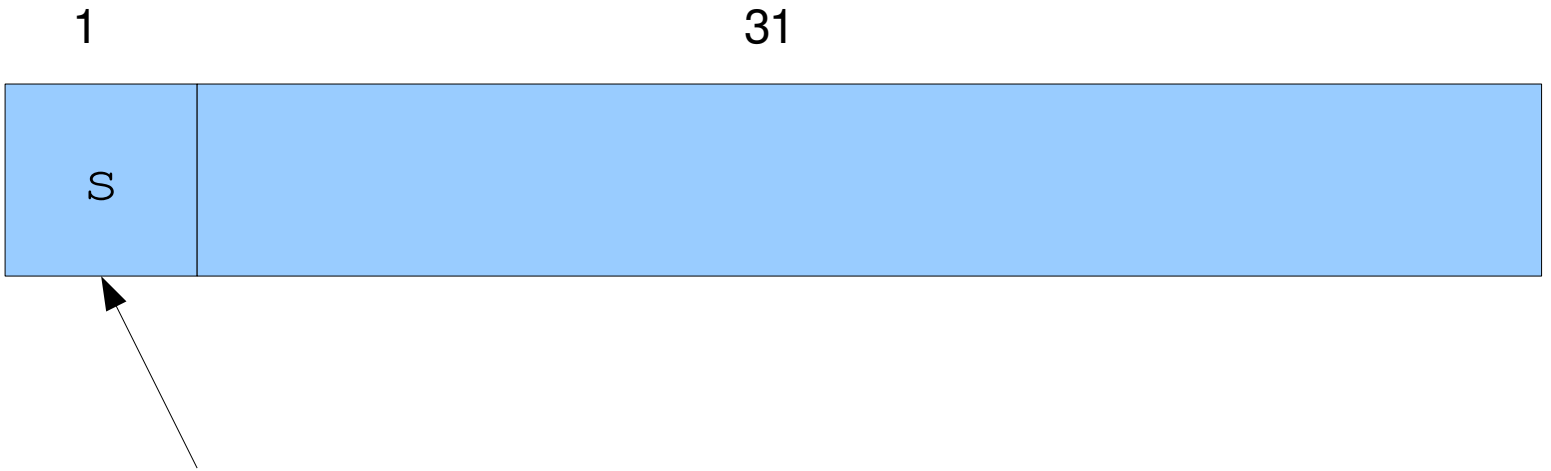
- Like a circuit, you can do “parallel” calculations



Back to the example

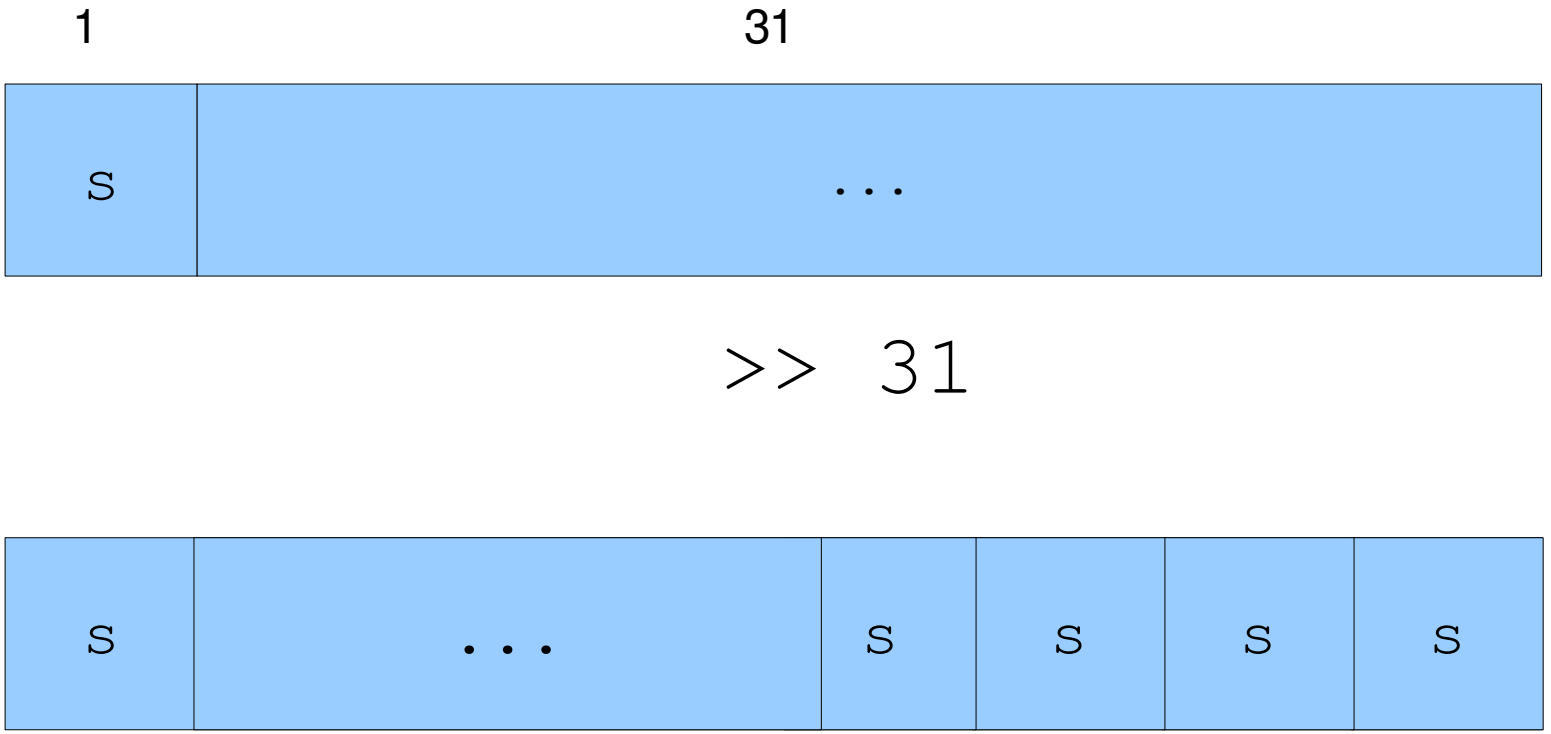
```
/*
 * isNegative - returns 1 if x is negative
 * Examples: isNegative(5) = 0, isNegative(-7) = 1
 * Legal ops: ! ~ & ^ | + << >>
 * Max ops: 5
 * Rating: 1
 */
int isNegative(int x) {
  return 2;
}
```

How can we tell whether an integer is negative?

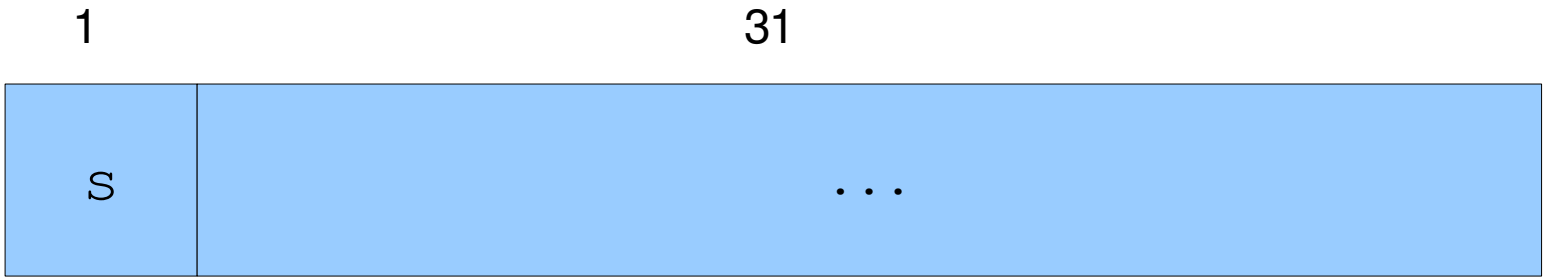


First bit of int tells us whether it is negative

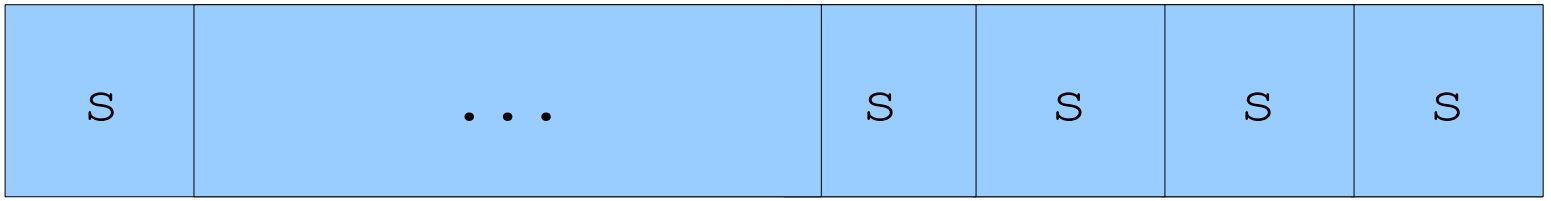
Now what?



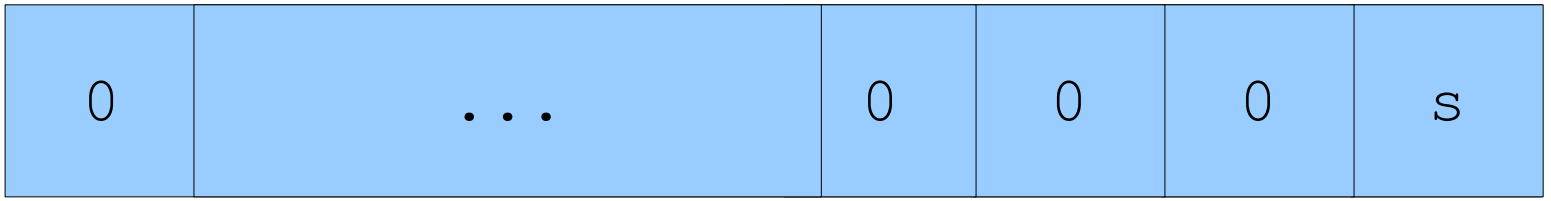
Right shift by 31 so that
top bit propagates to bottom



>> 31



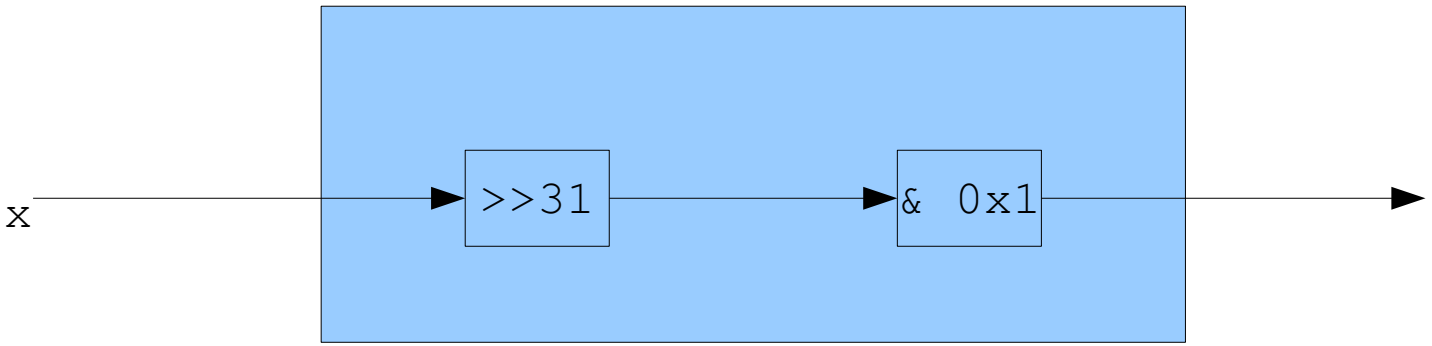
& 0x1



“Mask” the bottom bit so we get a result

Solution

```
/*
 * isNegative - returns 1 if x is negative
 * Examples: isNegative(5) = 0, isNegative(-7) = 1
 * Legal ops: ! ~ & ^ | + << >>
 * Max ops: 5
 * Rating: 1
 */
int isNegative(int x) {
  return (x >> 31) & 0x1;
}
```



Solution

```
/*
 * isNegative - returns 1 if x is negative
 * Examples: isNegative(5) = 0, isNegative(-7) = 1
 * Legal ops: ! ~ & ^ | + << >>
 * Max ops: 5
 * Rating: 1
 */
int isNegative(int x) {
  return (x >> 31) & 0x1;
}
```

What if we wanted isPositive()?

Solution

```
/*
 * isNegative - returns 1 if x is negative
 * Examples: isNegative(5) = 0, isNegative(-7) = 1
 * Legal ops: ! ~ & ^ | + << >>
 * Max ops: 5
 * Rating: 1
 */
int isPositive(int x) {
  return (x >> 31) + 1;
}
```

datalab example 2

```
/*
 * bitParity - returns 1 if x contains an odd
number of 0's
 * Examples: bitParity(5) = 0, bitParity(7) = 1
 * Legal ops: ! ~ & ^ | + << >>
 * Max ops: 20
 * Rating: 4
 */
int bitParity(int x) {
  return 2;
}
```

Simple Solution

```
/*
 * bitParity - returns 1 if x contains an odd
number of 0's
 * Examples: bitParity(5) = 0, bitParity(7) = 1
 * Legal ops: ! ~ & ^ | + << >>
 * Max ops: 20
 * Rating: 4
 */
int bitParity(int x) {
  int i, numBits = 0;

  for (i=0; i<32; i++) {
    numBits = numBits + (x & 0x1);
    x = x >> 1;
  }

  return numBits % 2;
}
```

More Elegant Solution

```
/*
 * bitParity - returns 1 if x contains an odd
number of 0's
 * Examples: bitParity(5) = 0, bitParity(7) = 1
 * Legal ops: ! ~ & ^ | + << >>
 * Max ops: 20
 * Rating: 4
 */
int bitParity(int x) {
  int i, parity = 0;

  for (i=0; i<32; i++) {
    parity = parity ^ (x & 0x1);
    x = x >> 1;
  }

  return parity;
}
```

Straight Line Solution

```
/*
 * bitParity - returns 1 if x contains an odd
number of 0's
 * Examples: bitParity(5) = 0, bitParity(7) = 1
 * Legal ops: ! ~ & ^ | + << >>
 * Max ops: 20
 * Rating: 4
 */
int bitParity(int x) {
  int parity = x & 0x1;
  parity = parity ^ ((x >> 1) & 0x1);
  parity = parity ^ ((x >> 2) & 0x1);
  ...
  parity = parity ^ ((x >> 30) & 0x1);
  parity = parity ^ ((x >> 31) & 0x1);

  return parity;
}
```

Best Solution

```
/*
 * bitParity - returns 1 if x contains an odd
number of 0's
 * Examples: bitParity(5) = 0, bitParity(7) = 1
 * Legal ops: ! ~ & ^ | + << >>
 * Max ops: 20
 * Rating: 4
 */
int bitParity(int x) {
  int parity16 = x ^ (x >> 16);
  int parity8 = parity16 ^ (parity16 >> 8);
  int parity4 = parity8 ^ (parity8 >> 4);
  int parity2 = parity4 ^ (parity4 >> 2);
  int parity = parity2 ^ (parity2 >> 1);

  return parity;
}
```

NVIDIA Interview Question

What does the following expression do?

```
! (x & (x - 1))
```