-for a structure **foo** on the runtime stack, use **foo.a** to access element **a**

-for a structure **foo** on the heap (**foo** is a structure pointer), use **foo->a** to access element **a**

- **struct al\* foo;**

**foo = (struct al\*)malloc(1 \* sizeof(struct al));**

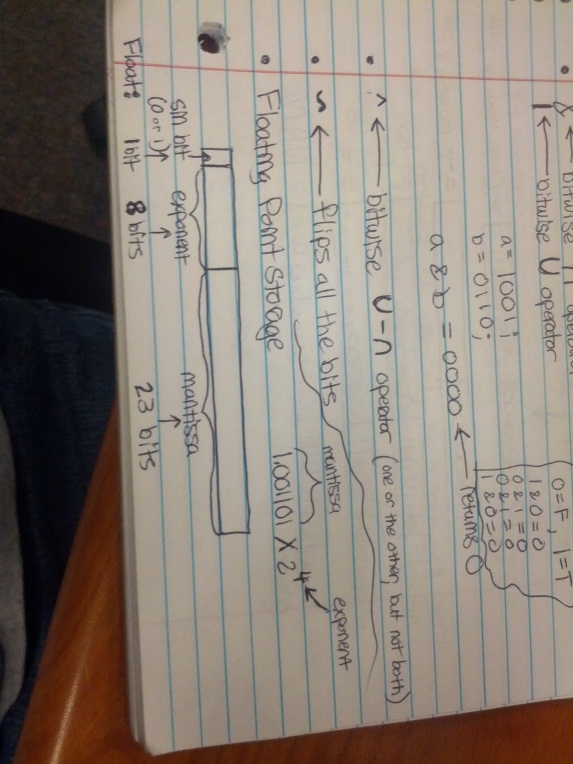
-you can’t return an array, because it’s declared on the runtime stack and is deleted when method ends

-pass an argument as **const** and you won’t be able to change its contents in the method (even a pointer)

- **int\* a;**

**\*a = s;** <------------ segfault, because you created the pointer, but didn’t allocate memory for it to point to

-most significant bit is bit on far left, least significant bit is bit on far right

****-1’s complement – flip 0’s and 1’s to get negative #’s (0 = 00000000, -0 = 11111111)

-2’s complement – take positive #, take 1’s complement, add 1 to get negative version (-1 = 11111111)

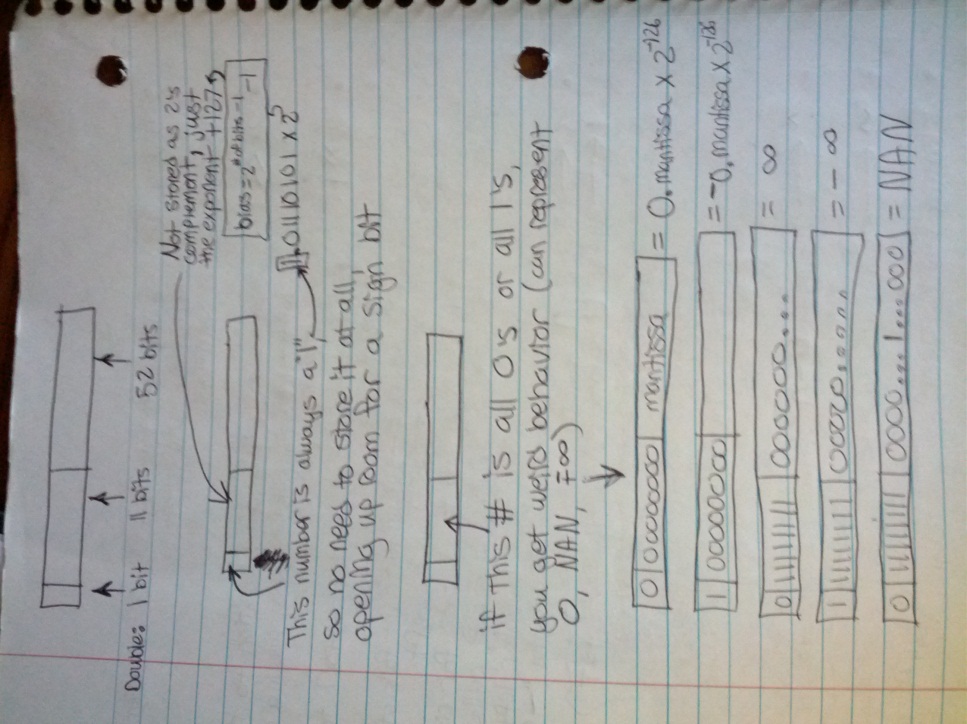
-computer doesn’t subtract, just adds 2’s complement

- **a >> 1;** <-------------- if signed negative #, bit on far right falls off, pulls in a 1 on far left to preserve negative status

- **^** <---------------- one or the other, but not both

- **~** <---------------- flips all the bits

-bias = 2# of bits – 1 – 1 (subtract this from the unsigned exponent #)

****

-**ServerSocket ss = new ServerSocket(7000);**

**-Socket receiver = ss.accept(); //starts listening on the port for a connection**

**-Socket sender = new Socket(“ip addr”, port #)**

- **MyThread mt = new MyThread();**

**Thread t = new Thread(mt);**

**threads.add(t);**

**t.start();**

**threads.get(i).join();**

**- public static synchronized add(int i)** <----- 1 thread at a time can run

**-signal(SIGCHLD, handler)**

**-int status;**

**int result = waitpid(pid, &status, 0);**

replacing pid with -1 makes parent start waiting for any process to

end (ends immediately if no children running)

replacing 0 with WNOHANG checks for dead child, but doesn’t wait

-**int rc = execvp(args[0], args);** <------ runs command at args[0],

but forks off and doesn’t return unless it fails (returns -1), so

fork off, run command in child, and **waitpid(pid, &reapingInfo, 0)**

in parent thread to reap it when finished

-**int pid = fork();** <--------------returns 0 in child, PID in parent

-**pthread\_t thread;**  <------- creates thread

-**pthread\_t threadHandles[threadCount];**  <----------- creates array of threads

**-pthread\_create(memory addr of pthread\_t, NULL, function name to call, (void\*)params);** <-------- starts thread

The function signature must look like this ------> **void\* function(void\* params)**

-**pthread\_join(thread, NULL);** <------ joins thread

-**pthread\_mutex\_t lock;** <--------------- creates lock

-**pthread\_mutex\_lock(&lock)** or **pthread\_mutex\_unlock(&lock)** <------------ locks/unlocks operation

-**omp\_get\_num\_threads()**, **omp\_get\_thread\_num()**

**-#pragma omp parallel for schedule(static, 5) reduction(+:sum) private(some\_unshared\_variable) nowait**

**-#pragma omp single {}**  <--------- only 1 thread can access this

-**#pragma omp critical {}** <--------- same as locking

- in C++, putting a **~** before a method name makes it a destructor

-copy constructor (a constructor that takes in a pointer to the same object)

-**Rational::Rational(Rational& r) {** <------------ pass by reference (adding “const” here prevents mutation of original object)

**this->num = r.num;**

**this->dem = r.dem; }**

-to give method access to private variables, add function signature to .h file and prefix it with “friend”

// Socket pointer

int sockfd;

// Get a socket of the right type

int sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if (sockfd < 0) {

printf("ERROR opening socket");

exit(1);

}

// port number

int portno = 7000;

// server address structure

struct sockaddr\_in serv\_addr;

// Set all the values in the server address to 0

memset(&serv\_addr, '0', sizeof(serv\_addr));

// Setup the type of socket (internet vs filesystem)

serv\_addr.sin\_family = AF\_INET;

// Basically the machine we are on...

serv\_addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

// Setup the port number

// htons - is host to network byte order

// network byte order is most sig bype first

// which might be host or might not be

serv\_addr.sin\_port = htons(portno);

// Bind the socket to the given port

if (bind(sockfd, (struct sockaddr \*) &serv\_addr, sizeof(serv\_addr)) < 0) {

printf("ERROR on binding\n");

exit(1);

}

// set it up to listen

listen(sockfd,5);

int newsockfd;

struct sockaddr\_in cli\_addr;

socklen\_t clilen = sizeof(cli\_addr);

// Wait for a call

newsockfd = accept(sockfd, (struct sockaddr \*) &cli\_addr, &clilen);

if (newsockfd < 0) {

printf("ERROR on accept");

exit(1);

}

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if (sockfd < 0) {

fprintf(stderr,"ERROR opening socket\n");

exit(0);

}

// port number

int portno = 7000;

// server address structure

struct sockaddr\_in serv\_addr;

// Set all the values in the server address to 0

memset(&serv\_addr, '0', sizeof(serv\_addr));

// Setup the type of socket (internet vs filesystem)

serv\_addr.sin\_family = AF\_INET;

// Setup the port number

// htons - is host to network byte order

// network byte order is most sig bype first

// which might be host or might not be

serv\_addr.sin\_port = htons(portno);

// Setup the server host address

struct hostent \*server;

server = gethostbyname("localhost");

if (server == NULL) {

fprintf(stderr,"ERROR, no such host\n");

exit(0);

}

memcpy(&serv\_addr.sin\_addr.s\_addr, server->h\_addr, server->h\_length);

// Connect to the server

if (connect(sockfd,(struct sockaddr \*) &serv\_addr, sizeof(serv\_addr)) < 0) {

printf("ERROR connecting\n");

exit(0);

}

- S = 2s Number of sets

E Number of lines per set

B = 2b Block size (bytes)

m = log2(M) Number of physical (main memory) address bits

M = 2m Max number of unique memory addresses

s = log2(S) Number of set index bits

b = log2(B) Number of block offset bits

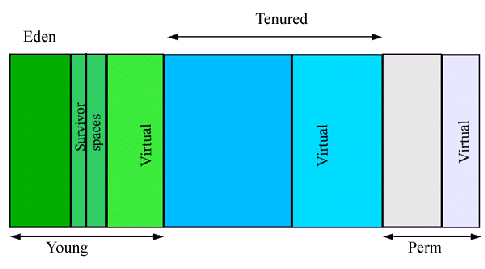
t = m – (s + b) Number of tag bits

C = B \* E \* S Cache size (bytes) including overhead (valid/tag bits)

**-bj: main.o Deck.o Card.o -g++ $(CCFILES) -MM > $(EXEC).d**

**g++ -o bj main.o Deck.o Card.o ifneq ($(MAKECMDGOALS), clean)**

**main.o: main.cc Deck.h Card.h include $(EXEC).d**

 **g++ -c -Wall main.cc endif**

**Deck.o: Deck.cc Deck.h Card.h**

**g++ -c -Wall Deck.cc**

**Card.o: Card.cc Card.h**

**g++ -c -Wall Card.cc**

**clean:**

**rm -f \*~**

**rm -f \*.o**

**rm -f bj**