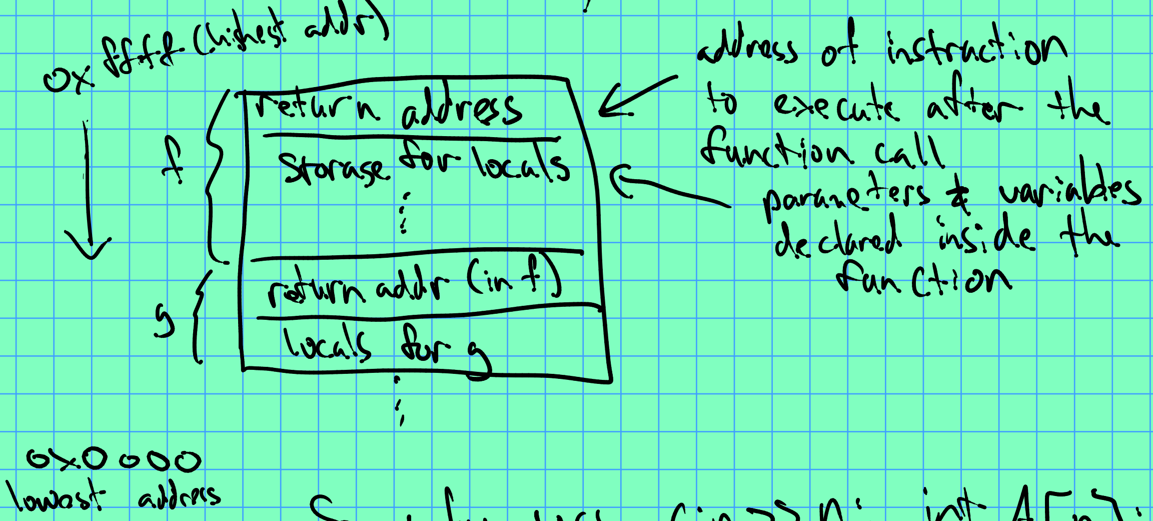
Dynamic Memory Management.

- Allocate memory as the program runs.
  Eg. push_back into a vector.

Historical notes: with old C++ standards, this wouldn't compile:

```
int main() {
    int n;
    cin >> n;
    int A[n];   // int A[100]; is fine,
      :         // since 100 is constant.
}
```

Recall what's on a typical stack frame:



0x ffff (highest addr)
return address
storage for locals
return addr (in f)
locals for g
f
g
address of instruction to execute after the function call
parameters & variables declared inside the function
0x0000 lowest address

So why was cin>>n; int A[n]; illegal? Well, compiler uses offsets from a frame pointer to refer to local variables & parameters.

The point: these offsets had to be known at compile time!

C++

```
int x;
x = 27;
```

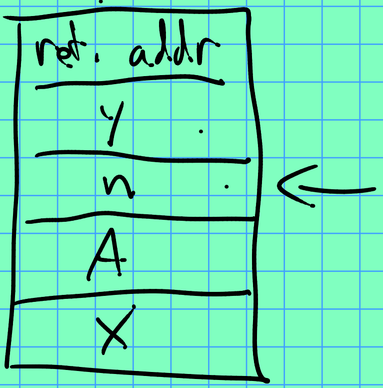
Assembly

```
// x associated w/ SP - 8
mov [SP - 8] 27
```

Note: SP ≡ Stack Pointer. Keeps track of where the current function's stack frame is located.

```
int g(int y) {
    int n;
    cin >> n;
    int A[n];
    int x;
}
```

frame for g:



if A's size is not known at compile time, can't compute offset for x!