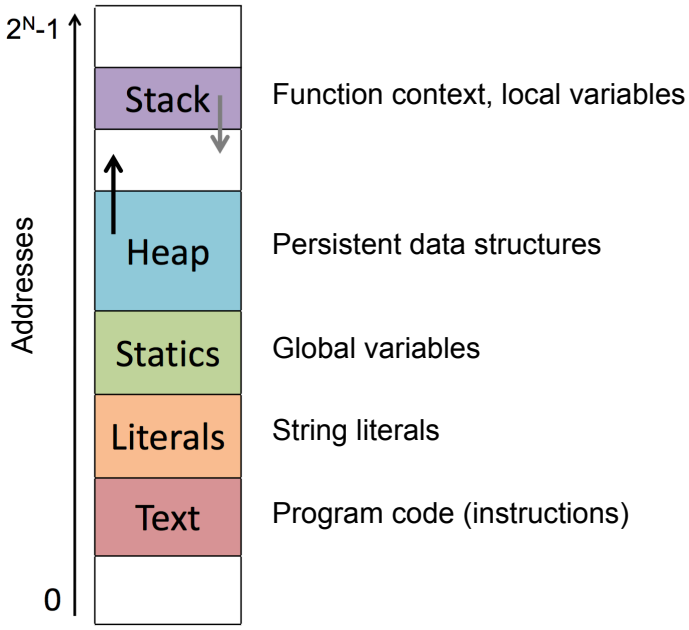
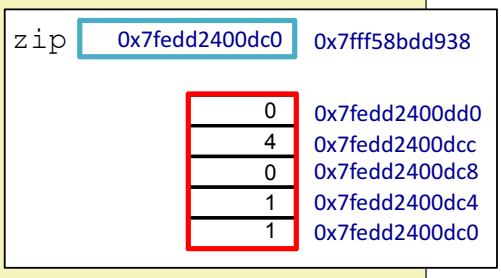
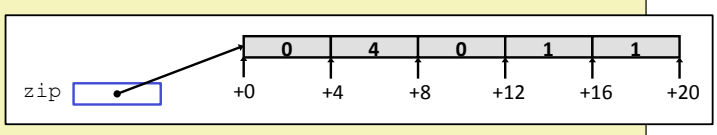
Memory Layout



Dynamic Memory Allocation Example

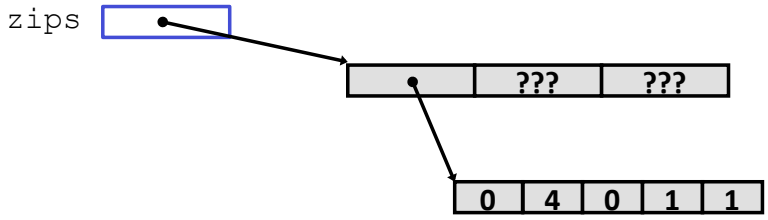
```
#define ZIP_LENGTH 5

int* zip = (int*) malloc(sizeof(int) * ZIP_LENGTH);
if (zip == NULL) {
    perror("malloc failed");
    exit(0);
}

zip[0] = 0;
zip[1] = 4;
zip[2] = 0;
zip[3] = 1;
zip[4] = 1;

printf("zip is");
for (int i = 0; i < ZIP_LENGTH; i++) {
    printf(" %d", zip[i]);
}
printf("\n");

free(zip);
```



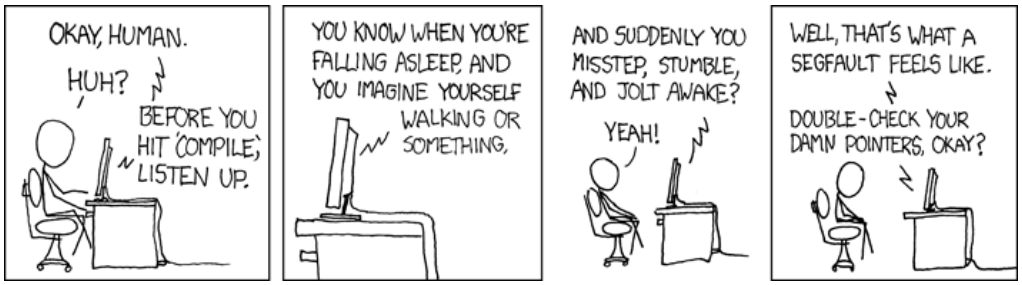


Pointers to Pointers to …

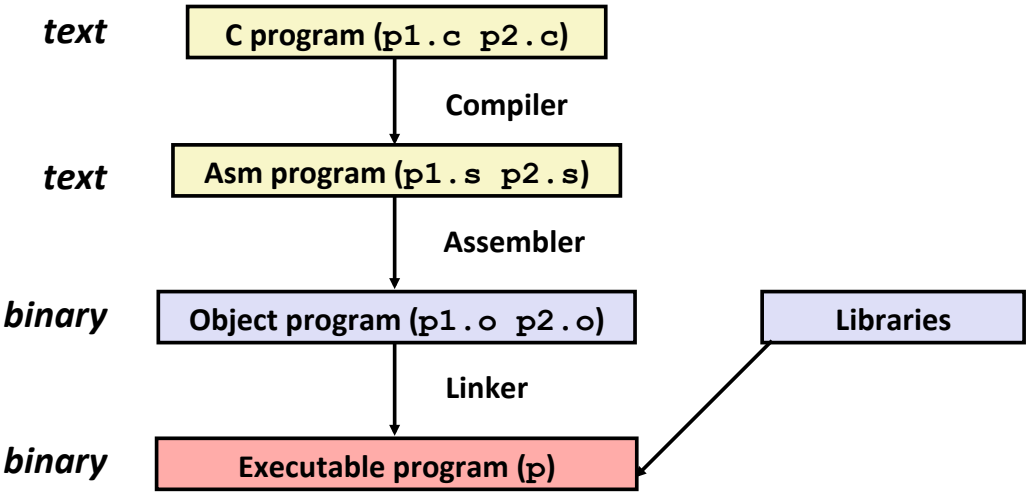
```
#define NUM_ZIPS 3

int** zips = (int**) malloc(sizeof(int*) * NUM_ZIPS);
...
zips[0] = (int*) malloc(sizeof(int) * ZIP_LENGTH);
...
int* first_zip = zips[0];
first_zip[0] = 0;
zips[0][1] = 4;
zips[0][2] = 0;
first_zip[3] = 1;
zips[0][4] = 1;
```



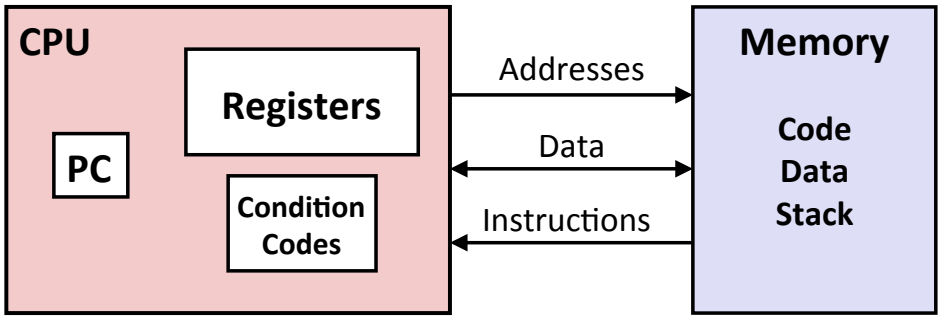
C Memory Errors



From C to Executable Code



Assembly View of the Machine



x86-64 Integer Registers

%rax	%eax	%r8	%r8d
%rbx	%ebx	%r9	%r9d
%rcx	%ecx	%r10	%r10d
%rdx	%edx	%r11	%r11d
%rsi	%esi	%r12	%r12d
%rdi	%edi	%r13	%r13d
%rsp	%esp	%r14	%r14d
%rbp	%ebp	%r15	%r15d

(not pictured: **%rip** = PC)