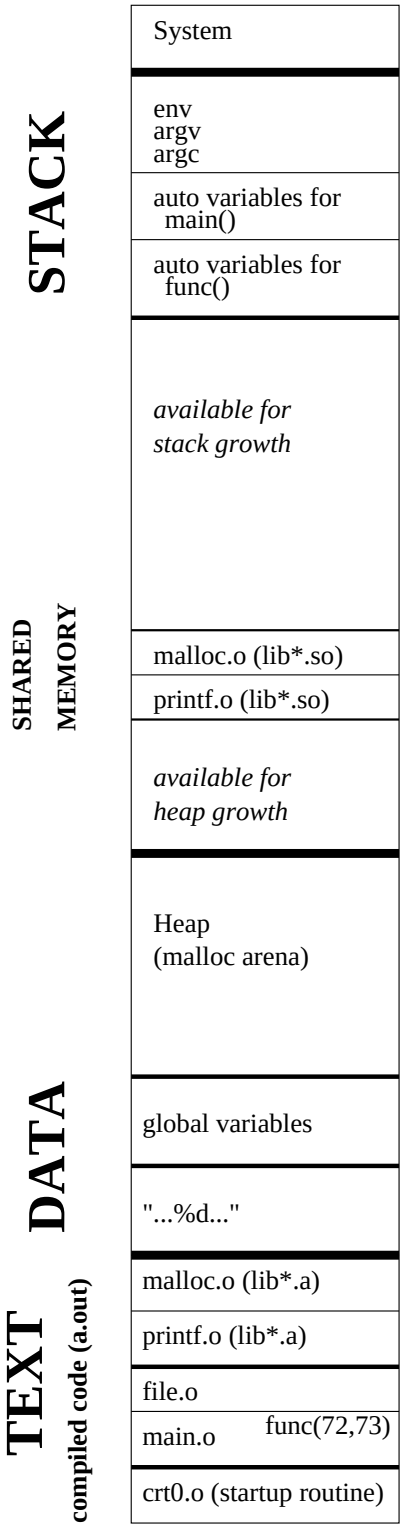
Memory Layout (Virtual address space of a C process)



High memory

← mfp − frame pointer (for main)

← stack pointer (grows downward if func() calls another function)

library functions if dynamically linked (usual case)

← brk point

uninitialized data (bss)

initialized data

library functions if statically linked (not usual case)

← ra (return address)

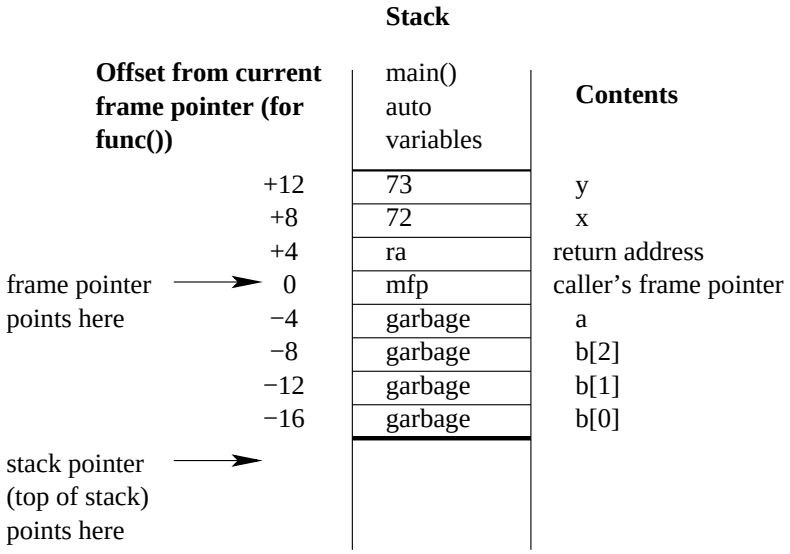
Low memory

Stack illustrated after the call func(72,73) called from main(), assuming func defined by:

```
func(int x, int y) {
   int a;
   int b[3];
 /* no other auto variables */
```

Assumes int = long = char * of size 4 and assumes stack at high address and descending down.

Expanded view of the stack



All auto variables and parameters are referenced via offsets from the frame pointer.

The frame pointer and stack pointer are in registers (for fast access).

When funct returns, the return value is stored in a register. The stack pointer is move to the y location, the code is jumped to the return address (ra), and the frame pointer is set to mfp (the stored value of the caller's frame pointer). The caller moves the return value to the right place.