Object file sections

Symbol table provides a mapping of symbols to addresses just like the one in your assembler. But it only contains things that:

- Might be needed by another file (globals and functions you define)
- You don’t have resolved (globals and functions others define that you need)

Relocation table holds addresses of _instructions_ that need to be fixed up with updated addresses. It contains fields that tell you:

- What address the instruction that needs to be fixed is at
- What type of instruction you have (lw, sw, jal, etc)
- What symbol you are using there

Statics

- Statics don't need to (but still can) go to the symbol table. We will accept both answers, so long as you are consistent about it.
- Instructions that use statics must still go to the relocation table.
- These statements are true for both global and local statics.

Instructions that need to be in the relocation table are anything that use a symbol in the object file’s symbol table or instructions that use static variables. Few example types:

- Calls to functions
- Uses of things in the data section (globals and statics)

Note: Statements that only declare variables do not translate to instructions in the binary. So there is nothing to relocate for such statements.

Where data goes

- All local variables go on stack. For arguments the first 4 go into registers. The rest go on stack.
- Dynamically allocated memory goes on the heap.
- Global, static variables, and constant strings go into data segment