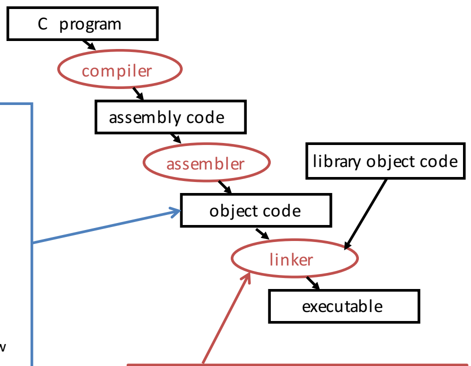
Linking and Loading

When your program is more than a single .asm file..

Linker: links separate object codes into one.



Object file header describes size and pieces of file.

Text segment contains machine code.

Static data contains long lived data.

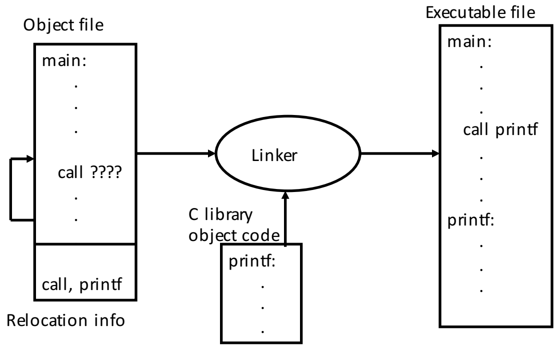
Relocation information identifies instructions and data that depend on absolute addresses.

Symbol table contains remaining labels such as external references.

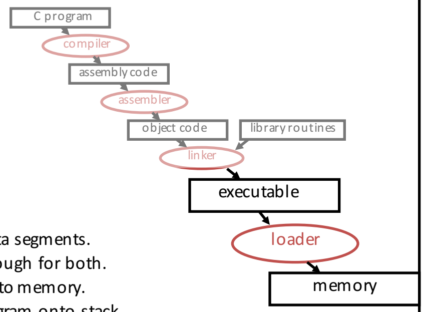
Debugging information concerning how modules were compiled.

1. Assign all text and data sections symbolic spots in address space.
2. Resolve symbolic labels to addresses.
3. Patch up absolute address references.

Linking schematic



Loader: loads executable into memory.



1. Determine size of text and data segments.
2. Create address space large enough for both.
3. Copy instructions and data into memory.
4. Copy parameters to main program onto stack.
5. Initialize registers (including $sp).
6. Jump to main procedure.

Static vs. dynamic linking

Update flexibility:

Static: Getting library updates requires relinking.

Dynamic**:** Library updates require no changes to executable.*

Code size:

Static: all code and static data that ***might*** ever be called or used

- must be included in the executable.
- must be loaded at run-time.

Dynamic**:** external code is linked and loaded only if used.

- may save static and dynamic work
- moves some static work to run-time.

*assuming the API does not change.

Dynamic linking (see board…)