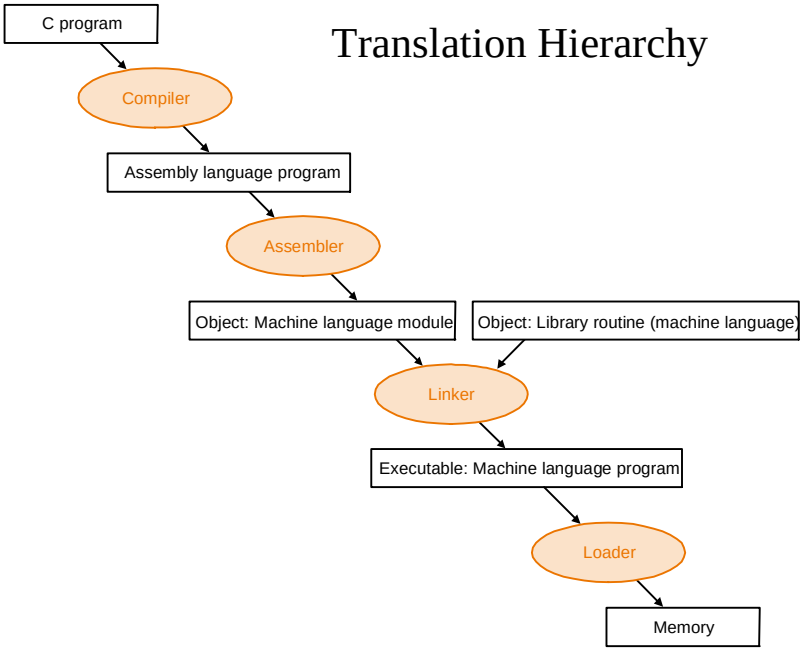
Assemblers, Linkers & Loaders

Translation Hierarchy



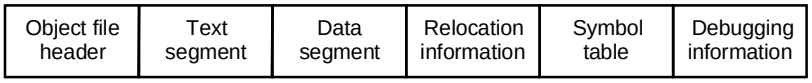
Translation Hierarchy

- Compiler
  - Translates high-level language program into assembly language (CS 440)
- Assembler
  - Converts assembly language programs into *object* files
    - Object files contain a combination of machine instructions, data, and information needed to place instructions properly in memory

Assemblers

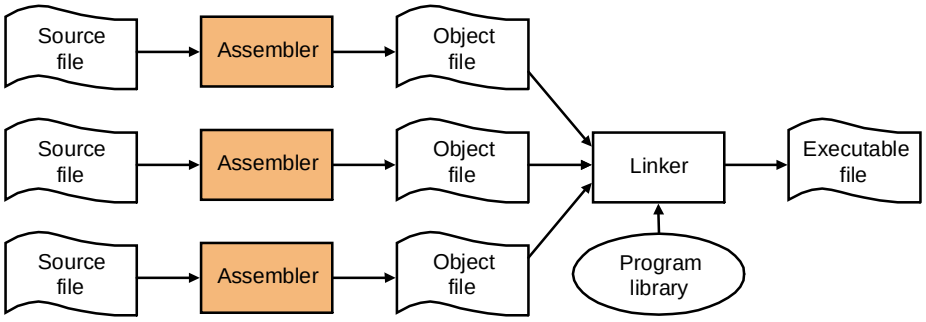
- Assemblers need to
  - translate assembly instructions and pseudo-instructions into machine instructions
  - Convert decimal numbers, etc. specified by programmer into binary
- Typically, assemblers make two passes over the assembly file
  - First pass: reads each line and records *labels* in a *symbol table*
  - Second pass: use info in symbol table to produce actual machine code for each line

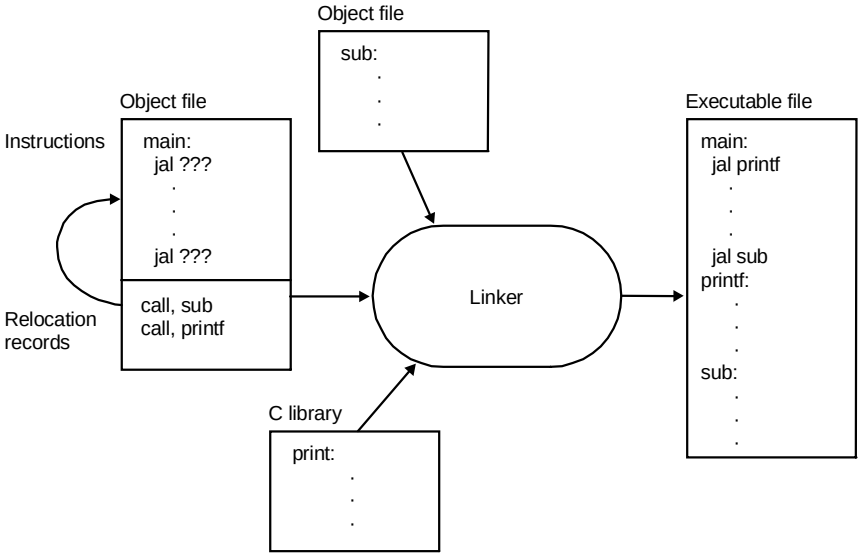
Object file format



- Object file header describes the size and position of the other pieces of the file
- Text segment contains the machine instructions
- Data segment contains binary representation of data in assembly file
- Relocation info identifies instructions and data that depend on absolute addresses
- Symbol table associates addresses with external labels and lists unresolved references
- Debugging info

Process for producing an executable file





Linker

- Tool that merges the object files produced by *separate compilation* or assembly and creates an executable file
- Three tasks
  - Searches the program to find library routines used by program, e.g. printf(), math routines,…
  - Determines the memory locations that code from each module will occupy and relocates its instructions by adjusting absolute references
  - Resolves references among files

Object file header			
	Name	Procedure A	
	Text size	100_{hex}	
	Data size	20_{hex}	
Text segment	Address	Instruction	
	0	lw $a0, 0($gp)	
	4	jal 0	
	...	...	
Data segment	0	(X)	
	...	...	
Relocation information	Address	Instruction type	Dependency
	0	lw	X
	4	jal	B
Symbol table	Label	Address	
	X	-	
	B	-	
Object file header			
	Name	Procedure B	
	Text size	200_{hex}	
	Data size	30_{hex}	
Text segment	Address	Instruction	
	0	sw $a1, 0($gp)	
	4	jal 0	
	...	...	
Data segment	0	(Y)	
	...	...	
Relocation information	Address	Instruction type	Dependency
	0	lw	Y
	4	jal	A
Symbol table	Label	Address	
	Y	-	
	A	-	

Executable file header		
	Text size	300_{hex}
	Data size	50_{hex}
Text segment	Address	Instruction
	$0040\ 0000_{hex}$	lw $a0, 8000_{hex}($gp
	$0040\ 0004_{hex}$	jal 40 0100_{he}
	...	...
	$0040\ 0100_{hex}$	sw $a1, 8020_{hex}($gp
	$0040\ 0104_{hex}$	jal 40 0000_{he}
	...	...
Data segment	Address	
	$1000\ 0000_{hex}$	(X)
	...	...
	$1000\ 0020_{hex}$	(Y)
	...	...

Loader

- Part of the OS that brings an executable file residing on disk into memory and starts it running
- Steps
  - Read executable file's header to determine the size of text and data segments
  - Create a new address space for the program
  - Copies instructions and data into address space
  - Copies arguments passed to the program on the stack
  - Initializes the machine registers including the stack ptr
  - Jumps to a startup routine that copies the program's arguments from the stack to registers and calls the program's main routine