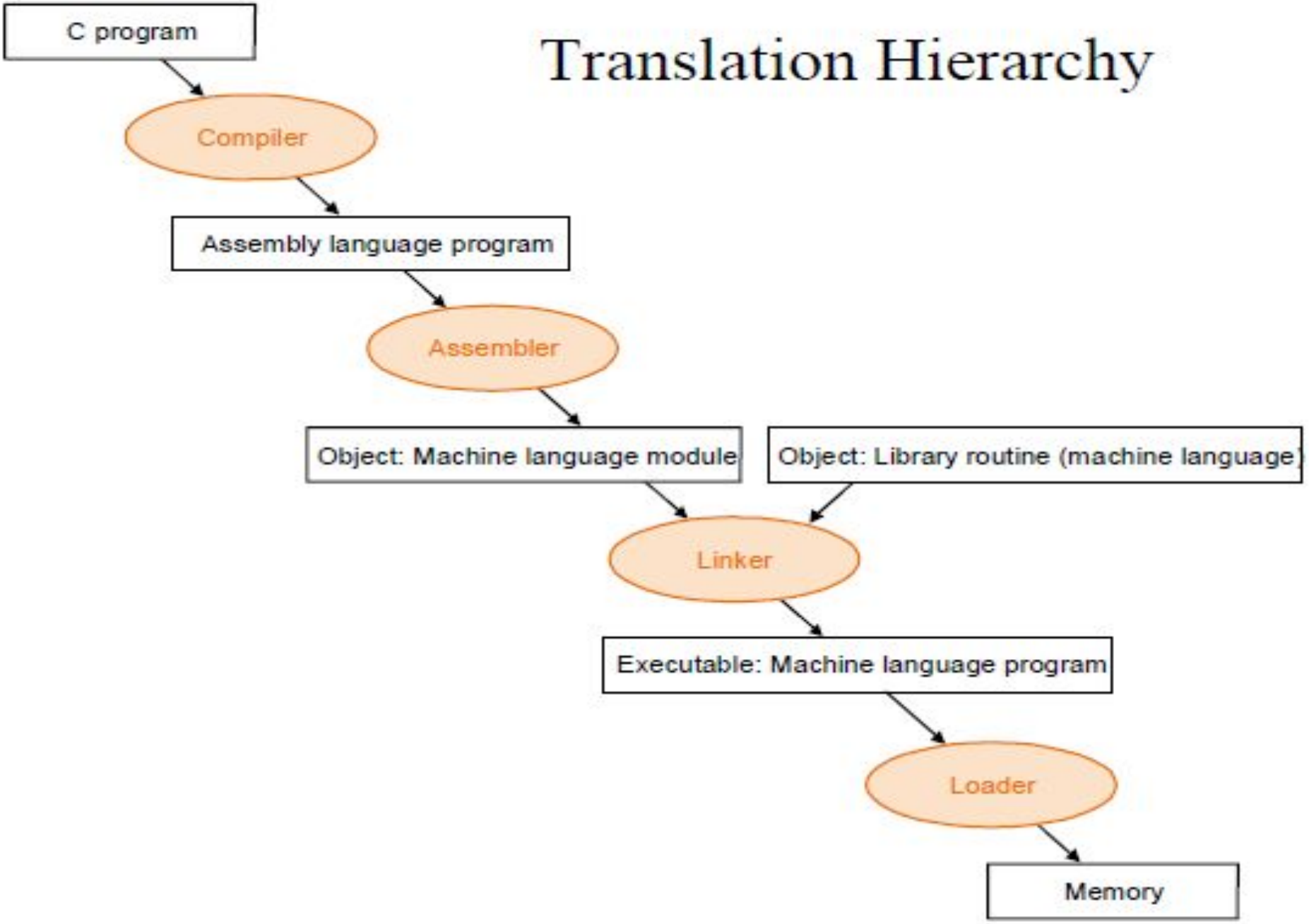
Linker

Presented By:
Sangeeta Ranjan

Translation Hierarchy



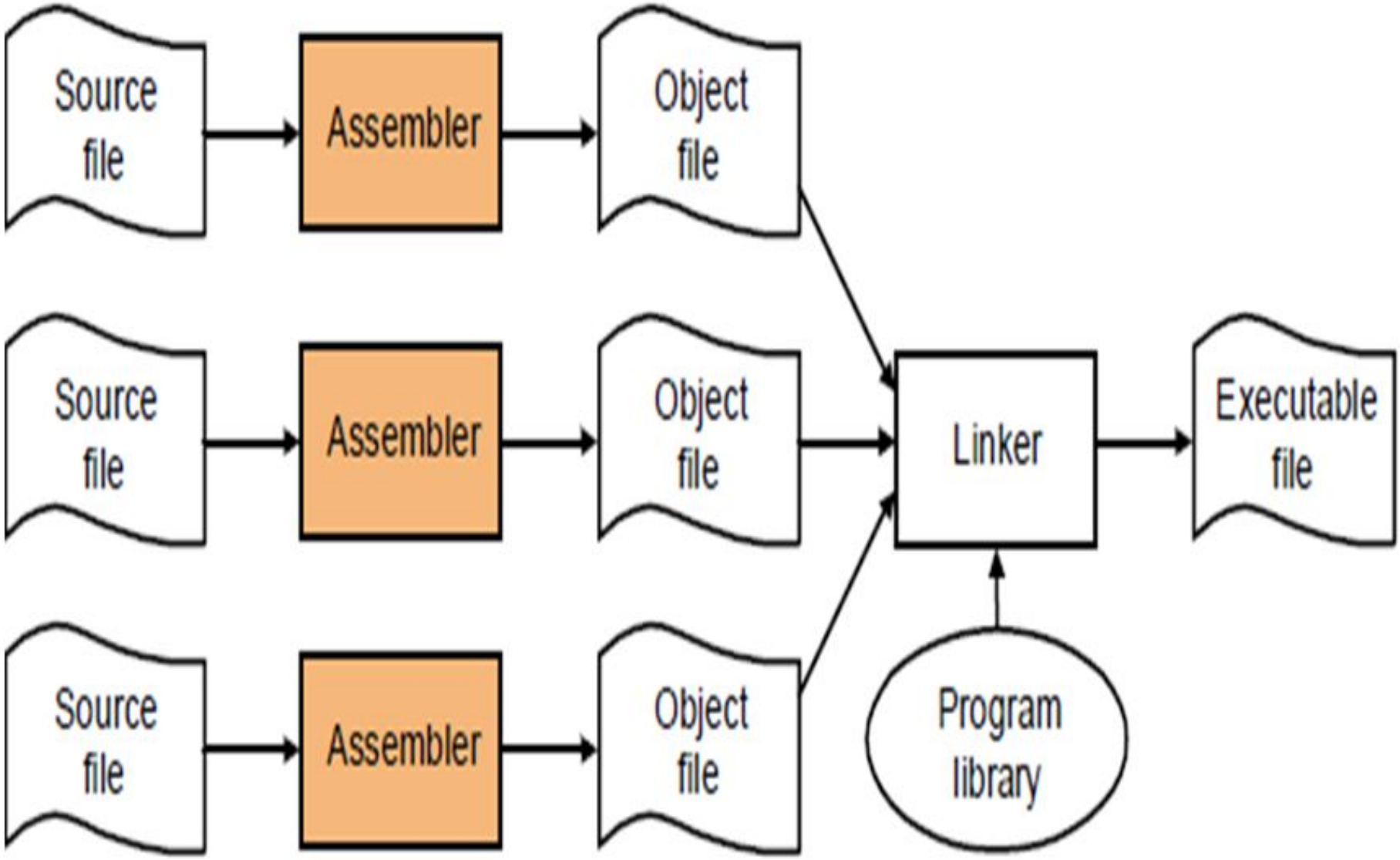
Linker

The Linker is the software program which binds many object modules to make a single object program.(Done Automatically when program is executed).

Three tasks:-

- Searches the program to find library routines used by program, e.g. Printf(), sqrt(), strcat() and various other.
- Determines the memory locations that code from each module will occupy and relocates its instructions by adjusting absolute references.

  Relocation, which modifies the object program so that it can be loaded at an address different from the location originally specified.
-


Source file
Assembler
Object file
Source file
Assembler
Object file
Linker
Executable file
Source file
Assembler
Object file
Program library

Relocation and linking concepts

- Execution of a program written in a language L involves following steps:
  - Translation of the program by translator of language L
  - Linking of the program with other program needed for its execution by a linker
  - Relocation of the program to execute from the specific memory area allocated to it by a linker
  - Loading of the program in the memory for the purpose of execution by a loader

Relocation and linking concepts

- Schematic of program execution

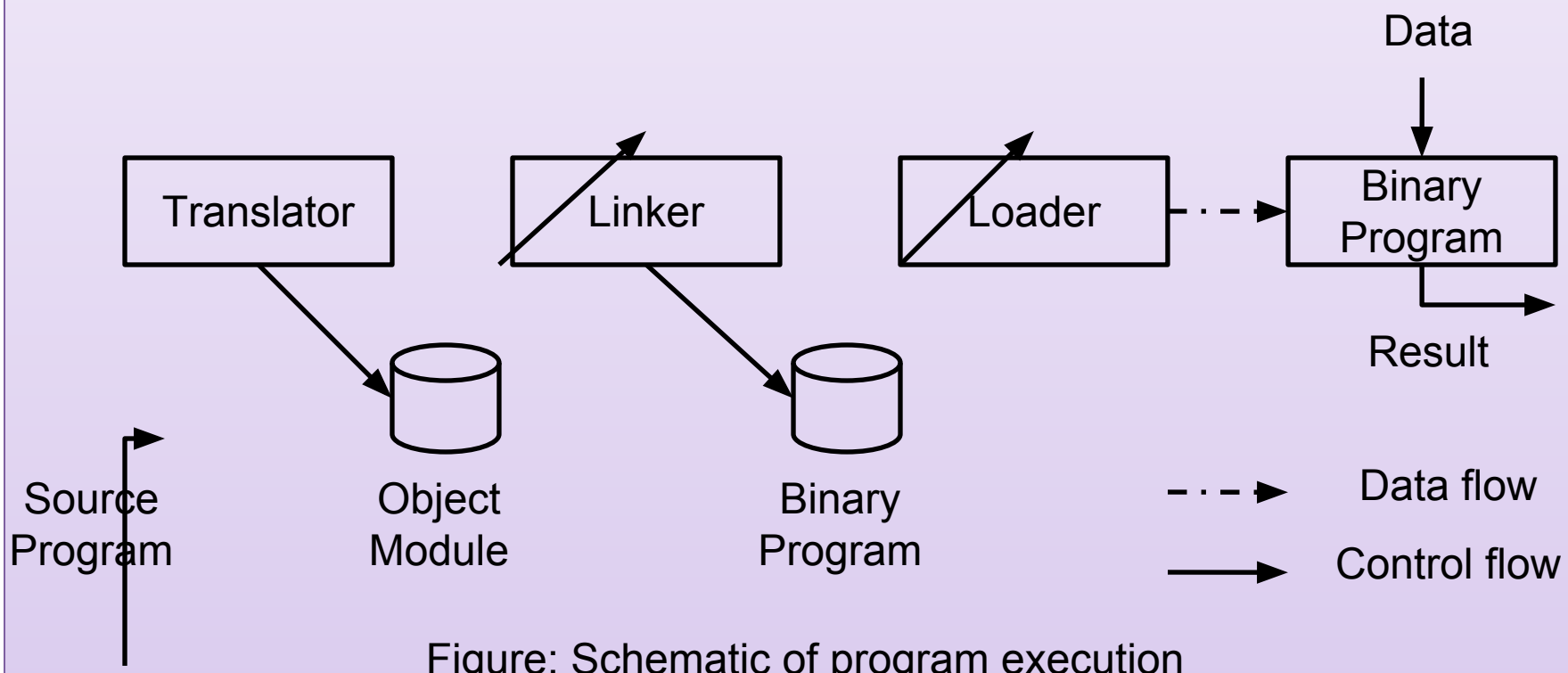


Figure: Schematic of program execution

Linking Concepts

- Computer programs typically comprise several parts or modules; all these parts/modules need not be contained within a single **object file**, and in such case refer to each other by means of **symbols**. Typically, an object file can contain three kinds of symbols:

✔ ***Publicly defined symbols***, which allow it to be called by other modules , also called as **public definition** .

✔ ***Externally defined symbols(***undefined symbols), which calls the other modules where these symbols are defined, also called as **external reference.**

Relocation and linking concepts

- Translated, linked and load time addresses
  - **Translated origin**: address of the origin assumed by translator which is specified by the programmer in an ORIGIN or START statement.
  - **Linked origin**: address of the origin assigned by the linker while Linking a binary program.
  - **Load time (or load) address** : address of the origin assigned by the loader while loading the program for execution.

Relocation and linking concepts

- Translated, linked and load time addresses example

	Statement		Address	Code
	START	500		
	EXTRN	MAX, ALPHA		
	READ	A	500)	+ 09 0 540
LOOP	.		501)	
	.			
	MOVER	AREG, ALPHA	518)	+ 04 1 000
	BC	ANY, MAX	519)	+ 06 6 000
	.			
	.			
	BC	LT, LOOP	538)	+ 06 1 501
	STOP		539)	+ 00 0 000
A	DS	1	540)	

✔ Translated origin of the program = 500
✔Translation time address of LOOP = 501
✔Suppose load time origin = 900
✔Load time address of LOOP = 901

Relocation and linking concepts

- Program Relocation
  - Program relocation is the process of modifying the addresses used in the address sensitive instructions of a program such that the program can execute correctly from the designated area of memory.
  - If linked origin ≠ translated origin, relocation must be performed by the linker.
  - If load origin ≠ linked origin, relocation must be performed by the loader.
  - If load origin = linked origin, such loaders are called absolute loader.

Relocation and linking concepts

- Program Relocation

	Statement		Address	Code
	START	500		
	ENTRY	TOTAL		
	EXTRN	MAX, ALPHA		
	READ	A	500)	+ 09 0 540
LOOP	.		501)	
	.			
	MOVER	AREG, ALPHA	518)	+ 04 1 000
	BC	ANY, MAX	519)	+ 06 6 000
	.			
	.			
	BC	LT, LOOP	538)	+ 06 1 501
	STOP		539)	+ 00 0 000

✔ Translated origin of the program = 500
✔Suppose link origin = 900
✔Relocation factor = 900-500 = 400

Relocation and linking concepts

- Linking
  - Linking is the process of binding an external reference to the correct link time address.
  - Program consist of
    - **Public definition –** a symbol may be referenced in other program unit. The ENTRY statement list the public definition of a program unit.
    - **External reference –** a reference to a symbol which is not defined in the program unit containing reference. The EXTRN statement lists the symbol to which external references are made in the program unit.

Relocation and linking concepts

- Linking

	Statement		Address	Code
	START	500		
	EXTRN	MAX, ALPHA		
	READ	A	500)	+ 09 0 540
LOOP	.		501)	
	.			
	MOVER	AREG, ALPHA	518)	+ 04 1 000
	BC	ANY, MAX	519)	+ 06 6 000
	.			
	.			
	BC	LT, LOOP	538)	+ 06 1 501
	STOP		539)	+ 00 0 000
A	DS	1	540)	

Program P

Relocation and linking concepts

- Linking

Program Q

	Statement		Address	Code
	START	200		
	ENTRY	ALPHA		

ALPHA	DS	25	231)	+ 00 0 025
	END			

  - Let the link origin of P be 900 and its size be 42 words.
  - The link origin of Q is therefore 942 and link time address of ALPHA is 973.
  - Linking is performed by putting link time address of ALPHA in the instruction of P using ALPHA.

Relocation and linking concepts

- Object Module
  - Object module of a program contains all information necessary to relocate and link the program with other program.
  - It consist of 4 components:
    1. **Header** : contains
       - Translated origin
       - Size
       - Execution start address of program P
    2. **Program** : contains machine language program corresponding to program P.

Relocation and linking concepts

- Object Module

3. **Relocation Table (RELOCTAB)** : describes IRRp (Set of instruction requiring relocation). It contains
   - Translated address of address sensitive instruction.

4. **Linking Table (LINKTAB)** : contains information concerning public definition and external references.

   This table contains
   - Symbol : Symbolic name.
   - Type : PD/EXT
   - Translated Address
     - ✔ For public definition, this is the address of the first memory word allocated to the symbol.
     - ✔ For external reference, it is address of the memory word which is required to contain the address of the symbol

Relocation and linking concepts

- Object Module

	Statement		Address	Code
	START	500		
	EXTRN	MAX, ALPHA		
	READ	A	500)	+ 09 0 540
LOOP	.		501)	
	.			
	MOVER	AREG, ALPHA	518)	+ 04 1 000
	BC	ANY, MAX	519)	+ 06 6 000
	.			
	.			
	BC	LT, LOOP	538)	+ 06 1 501
	STOP		539)	+ 00 0 000
A	DS	1	540)	

Program P

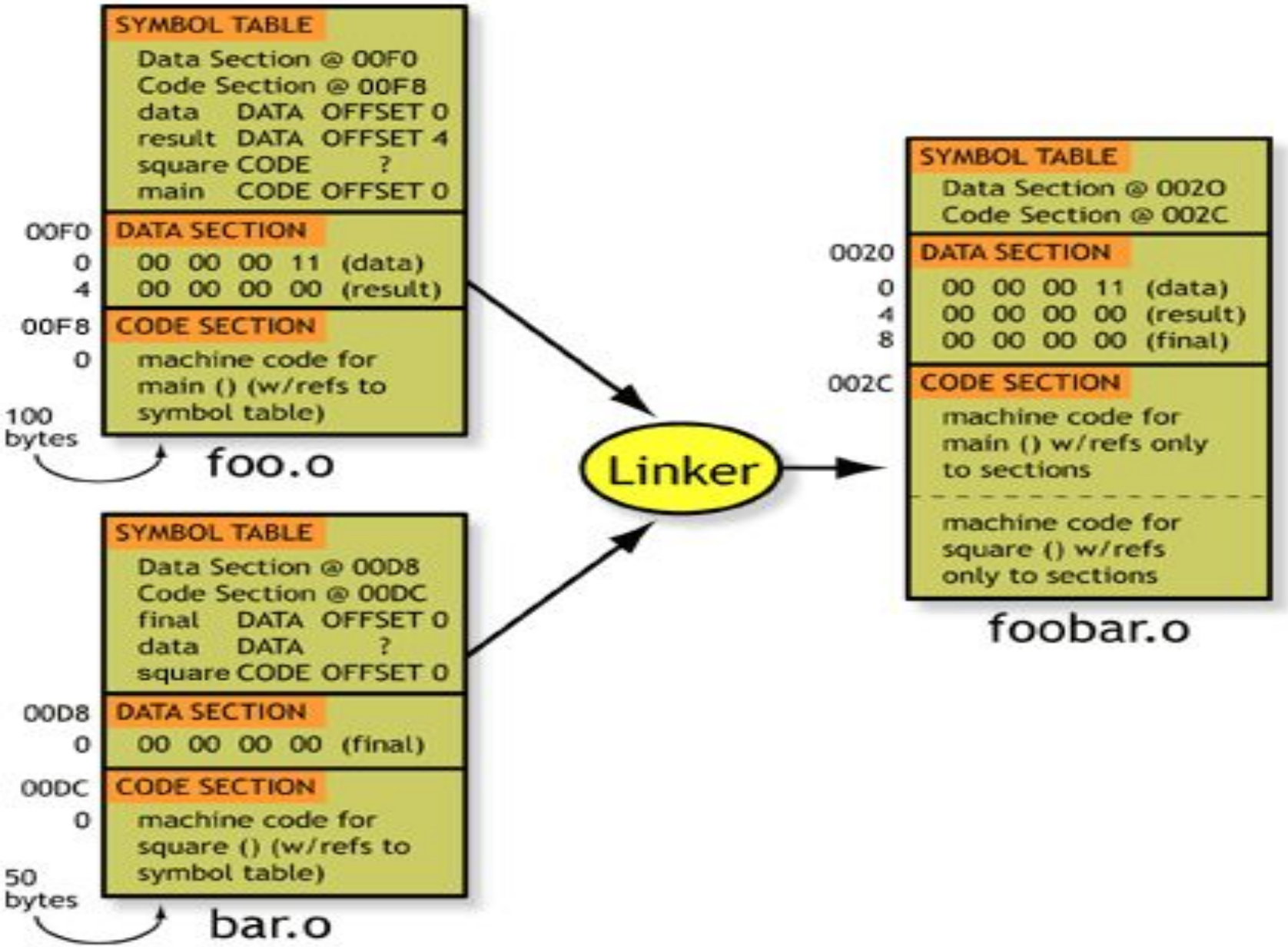
Relocation and linking concepts

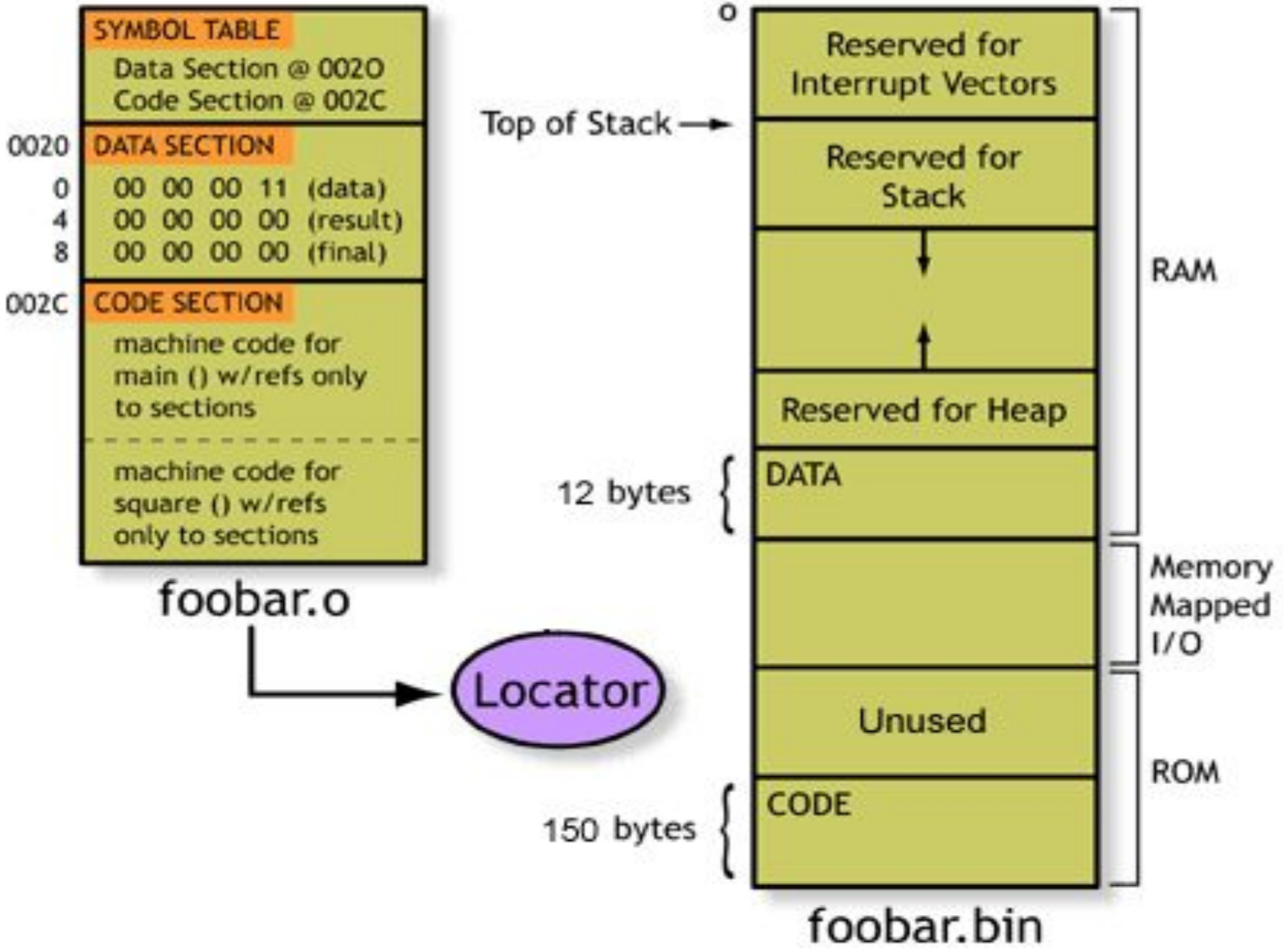
- Object Module
  - for program P
    1. Header : translated origin = 500, size = 42, execution start address = 500
    2. Machine language instruction shown in figure.
    3. Relocation table

	Translated address
	500
	538

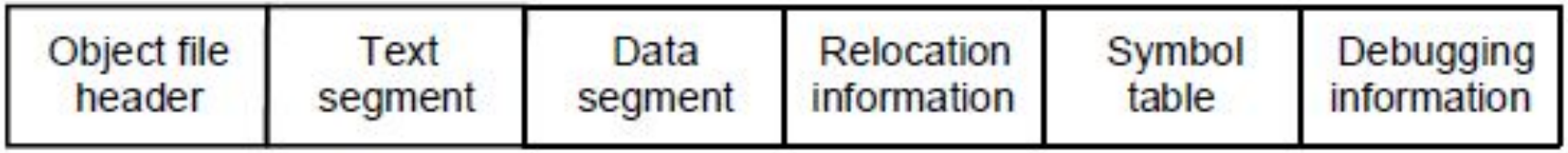
4. Linking table

Symbol	Type	Translated address
ALPHA	EXT	518
MAX	EXT	519


SYMBOL TABLE
Data Section @ 00F0
Code Section @ 00F8
data DATA OFFSET 0
result DATA OFFSET 4
square CODE ?
main CODE OFFSET 0
00F0
0
4
DATA SECTION
00 00 00 11 (data)
00 00 00 00 (result)
00F8
0
CODE SECTION
machine code for main () (w/refs to symbol table)
100 bytes
foo.o
SYMBOL TABLE
Data Section @ 00D8
Code Section @ 00DC
final DATA OFFSET 0
data DATA ?
square CODE OFFSET 0
00D8
0
DATA SECTION
00 00 00 00 (final)
00DC
0
CODE SECTION
machine code for square () (w/refs to symbol table)
50 bytes
bar.o
Linker
SYMBOL TABLE
Data Section @ 0020
Code Section @ 002C
0020
0
4
8
DATA SECTION
00 00 00 11 (data)
00 00 00 00 (result)
00 00 00 00 (final)
002C
CODE SECTION
machine code for main () w/refs only to sections
machine code for square () w/refs only to sections
foobar.o


SYMBOL TABLE
Data Section @ 0020
Code Section @ 002C
0020
DATA SECTION
0 00 00 00 11 (data)
4 00 00 00 00 (result)
8 00 00 00 00 (final)
002C
CODE SECTION
machine code for main () w/refs only to sections
machine code for square () w/refs only to sections
foobar.o
Locator
0
Top of Stack
Reserved for Interrupt Vectors
Reserved for Stack
Reserved for Heap
DATA
12 bytes
RAM
Memory Mapped I/O
Unused
CODE
150 bytes
ROM
foobar.bin

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

Static Linking

- Static linking is the process of copying all library modules used in the program into the final executable image.
- Statically linked files are significantly larger in size because external programs are built into the executable files.
- In static linking if any of the external programs has changed then they have to be recompiled and re-linked again else the changes won't reflect in existing executable file.
- Statically linked program takes constant load time every time it is loaded into the memory for execution.
- Programs that use statically-linked libraries are usually faster than those that use shared libraries.
- Disadvantages:

▪Behavior of executable files cannot change without re-linking them

▪Size of Program increases.

Dynamic linking

- In dynamic linking the names of the external libraries (shared libraries) are placed in the final executable file while the actual linking takes place at run time when both executable file and libraries are placed in the memory.
- Dynamic linking is performed at run time by the operating system.
- In dynamic linking only one copy of shared library is kept in memory. This

Dynamic linking

- Many operating system environments allow dynamic linking, that is the postponing of the resolving of some undefined symbols until a program is run.
- ***That means that the executable code still contains undefined symbols, plus a list of objects or libraries that will provide definitions for these. Loading the program will load these objects/libraries as well, and perform a final linking.***

Advantages and Disadvantages

- ***Advantages:***
- Often-used libraries (for example the standard system libraries) need to be stored in only one location, not duplicated in every single binary.
- If an error in a library function is corrected by replacing the library, all programs using it dynamically will benefit from the correction after restarting them. Programs that included this function by static linking would have to be re-linked first.

Thank You!