Chapter 6:
Loaders

Mrs. Sunita M Dol (Aher),
Assistant Professor,
Computer Science and Engineering Department,
Walchand Institute of Technology, Solapur, Maharashtra

6. Loaders

- Function of loader
- General loader scheme
- Absolute loader
- Relocating loader
- Direct linking loader
- Dynamic loading

6. Loaders

- Function of loader
- General loader scheme
- Absolute loader
- Relocating loader
- Direct linking loader
- Dynamic loading

Function of Loader

- Loader
  - The loader is a program which accepts the object program decks, prepare these program for execution by the computer and initiates the execution.

Function of Loader

- Loader

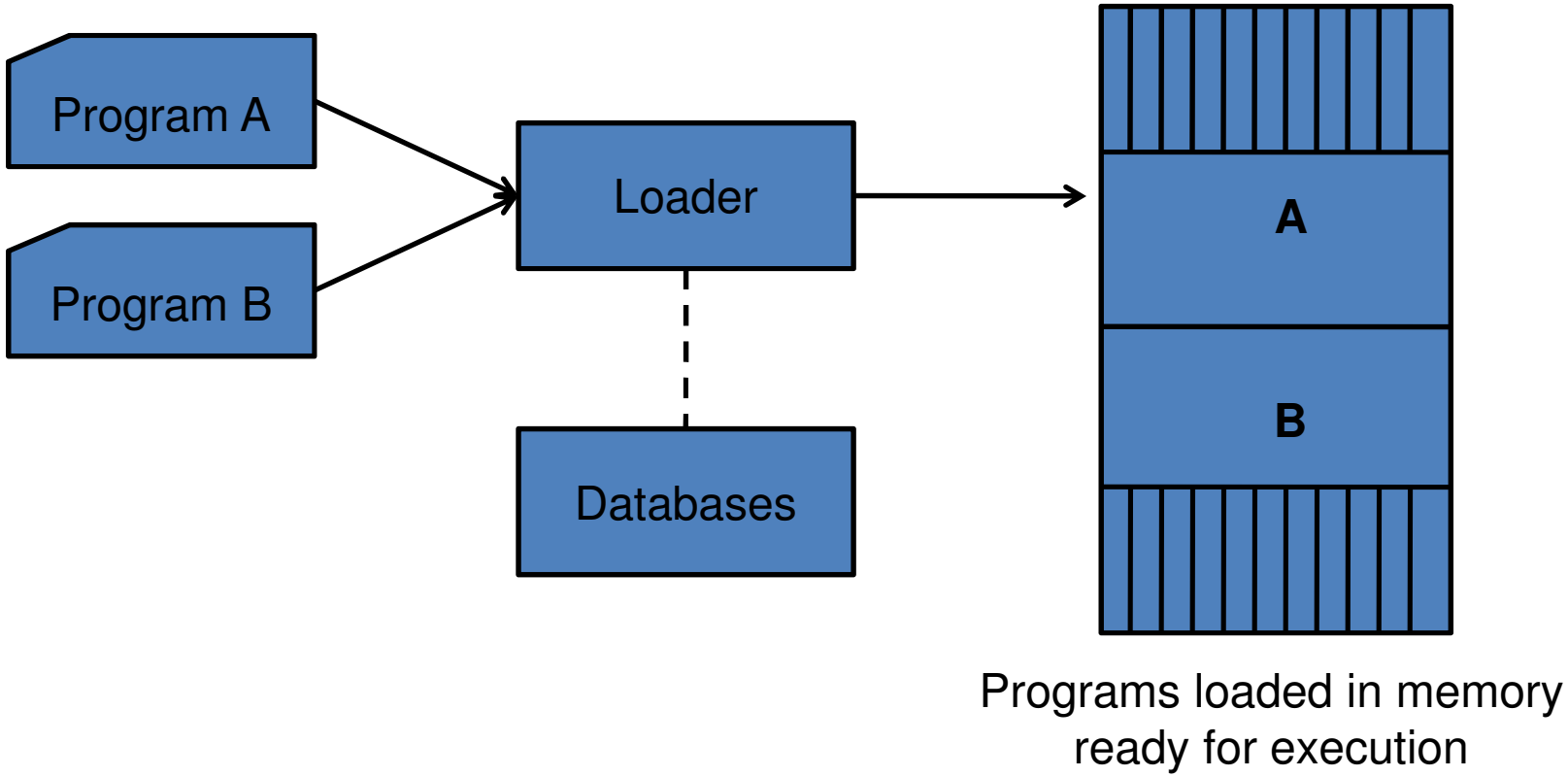


Figure: General loading scheme

Function of Loader

- Four Functions of Loader
  - Allocation : allocate space in memory for program.
  - Linking: resolve symbolic references between object decks
  - Relocation: adjust all address dependent locations such as address constants to corresponds to allocated space.
  - Loading : physically place the machine instruction and data into memory.

Loader Scheme

- Types of Loader
  - "Compile and Go" Loader
  - General Loader scheme
  - Absolute Loader
  - Relocating Loader
  - Direct Linking Loader
  - Dynamic Loader

“Compile and Go” Loader

- The assembler runs in one part of memory and it place the assembled machine instruction and data directly into their assigned memory locations.
- When the assembly is completed, the assembler causes a transfer to starting instruction of the program.
- The loader consist of one instruction that transfers to the starting instruction of newly assembled program.

“Compile and Go” Loader

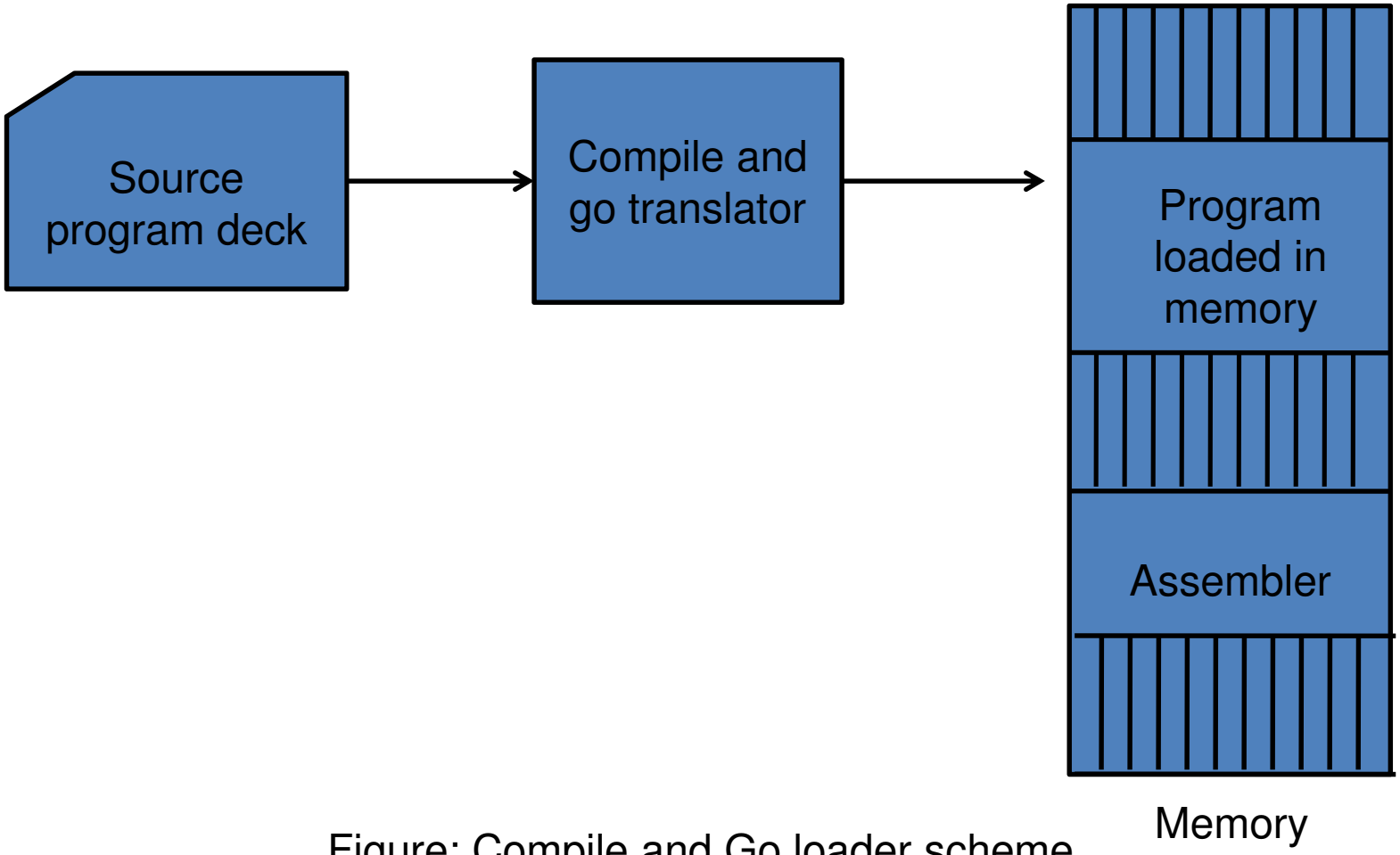


Figure: Compile and Go loader scheme

“Compile and Go” Loader

- Advantages
  - Easy to implement
  - No extra procedure

- Disadvantages
  - A portion of memory is wasted.
  - It is necessary to retranslate the user’s program deck every time it is run.
  - Difficult to handle multiple segment written in different languages.

6. Loaders

- Function of loader
- General loader scheme
- Absolute loader
- Relocating loader
- Direct linking loader
- Dynamic loading

General Loader Scheme

- The loader accepts the assembled machine instruction, data and other information present in object format and places machine instruction and data in core in an executable computer form.
- The loader is assumed to be smaller so that more memory is available to user.
- Reassembly is no longer necessary to run the program at later stage.

General Loader Scheme

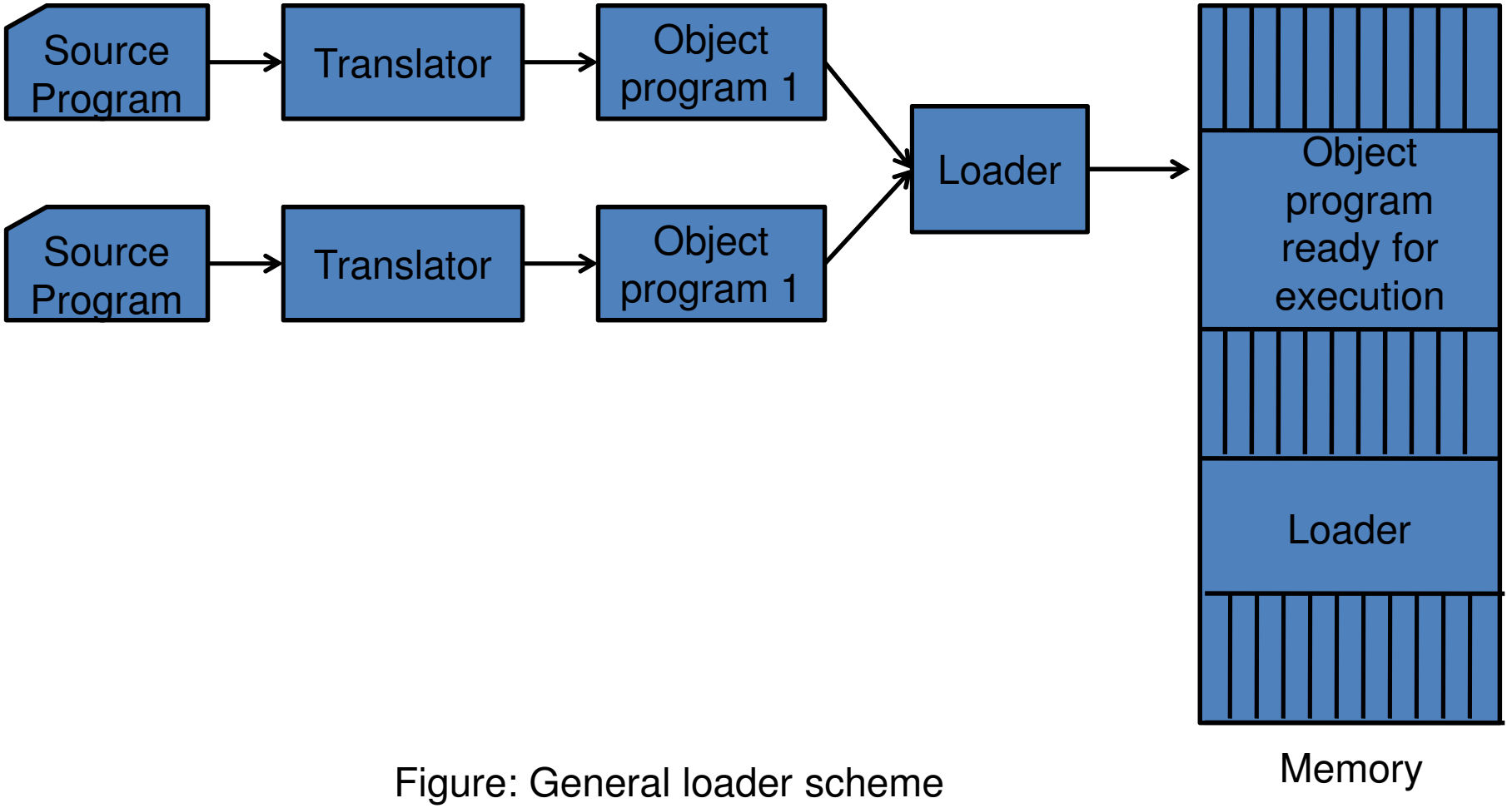


Figure: General loader scheme

6. Loaders

- Function of loader
- General loader scheme
- Absolute loader
- Relocating loader
- Direct linking loader
- Dynamic loading

Absolute Loader

- The assembler outputs the machine language translation of the source program.
- The data is punched on the cards instead of being placed directly in memory.
- The loader accepts the machine text and places it in the core at the location prescribed by the assembler.

Absolute Loader

	MAIN Program		Location		Instruction
MAIN	START	100			
	BALR	12,0	100	BALR	12,0
	USING	MAIN+2, 12	102	.	
	.		.	.	
	.		.	.	
	L	15, ASQRT	120	L	15, 142(0, 12)
	BALR	14,15	124	BALR	14, 15
	.		.	.	
	.		.	.	
ASQRT	DC	F'400'	244	F'400'	
	END		248		

Absolute Loader

	SQRT SUBROUTINE		Location		Instruction
SQRT	START	400	400		
	USING	*, 15	.	.	
	.		.	.	
	.		.	.	
	BR	14	476	BCR	15, 14

Absolute Loader

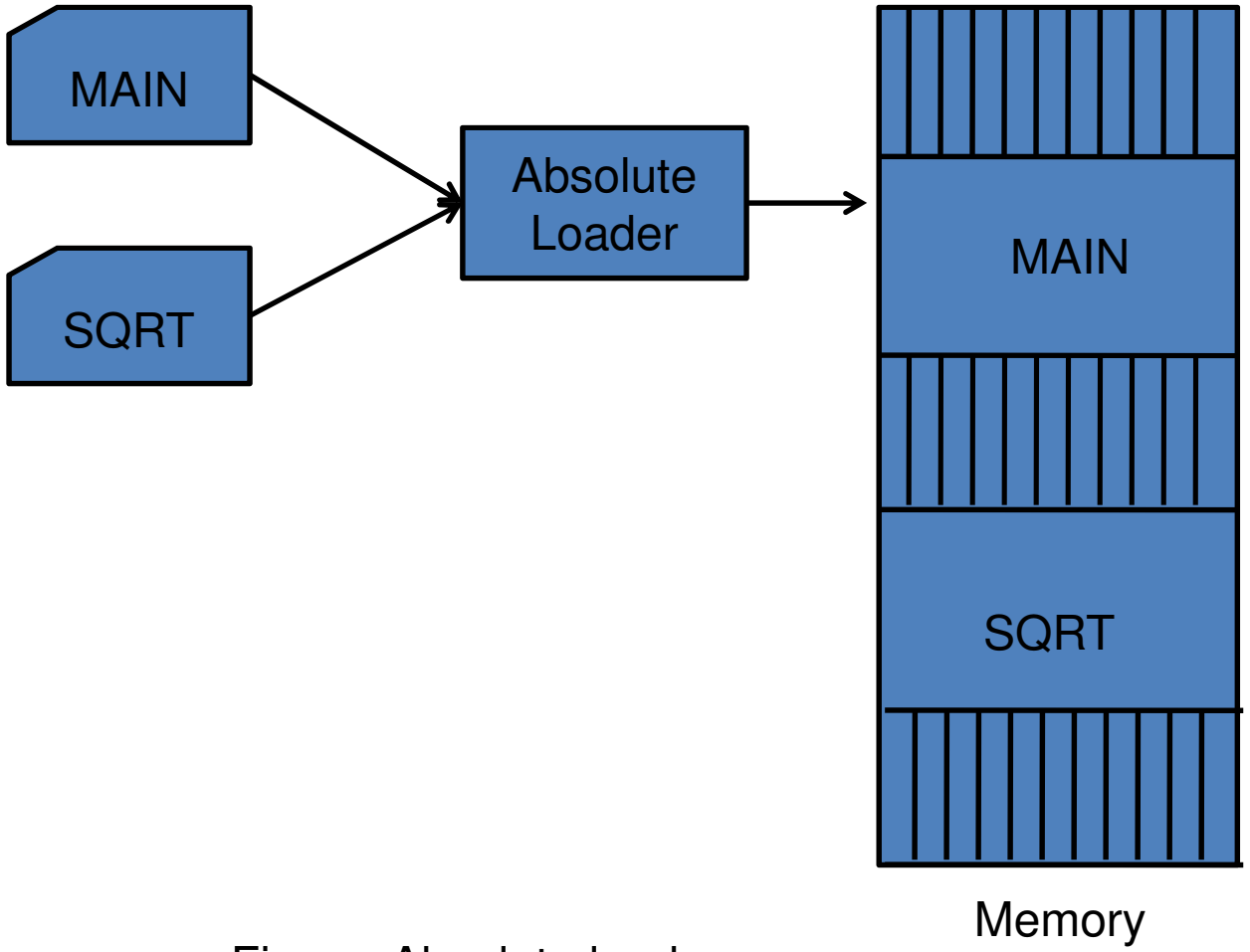


Figure: Absolute loader

Absolute Loader

- Advantages
  - Simple to implement
  - More core available to user

- Disadvantages
  - The programmer must specify to the assembler the address in core where the program is to be loaded.
  - If there are multiple subroutine then the programmer must remember address of each.

Absolute Loader

- Four loader function
  - Allocation by programmer
  - Linking by programmer
  - Relocation by assembler
  - Loading by loader

6. Loaders

- Function of loader
- General loader scheme
- Absolute loader
- Relocating loader
- Direct linking loader
- Dynamic loading

Relocating Loader

- The assembler assembles each procedure segment independently.
- The output of a relocating assembler is
  - the object program and information about all other programs it references
  - Relocation information i.e. the locations which are dependent on the core allocation.
- For each source program, the assemblers output a machine translation of the program prefixed by a transfer vector.

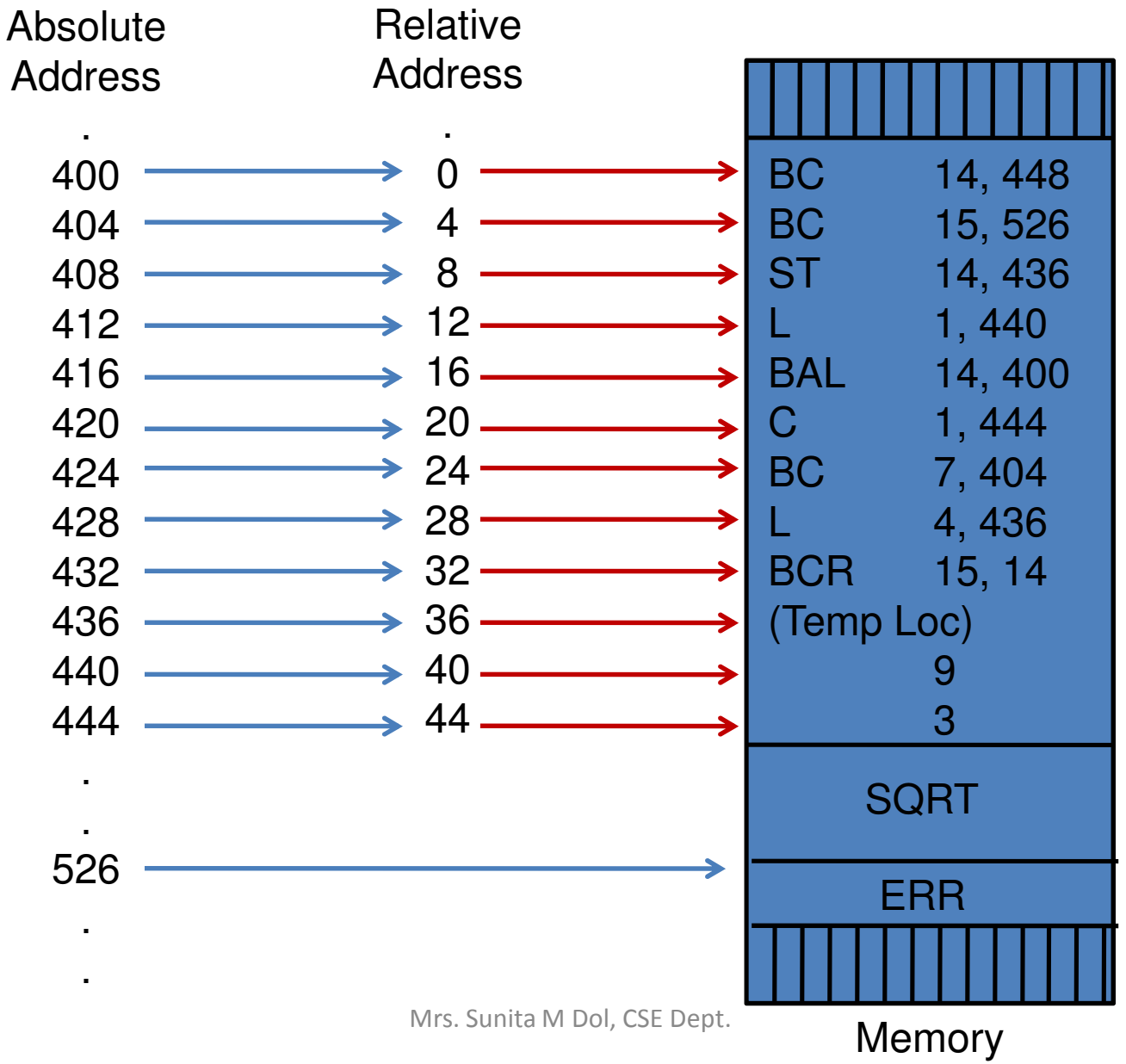
Relocating Loader

Program length = 48 bytes

Transfer vector = 8 bytes

Source Program			Rel. addr.	Relocation	Object code	
MAIN	START					
	EXTRN	SQRT	0	00	'SQRT'	
	EXTRN	ERR	4	00	'ERRb'	
	ST	14, SAVE	8	01	ST	14, 36
	L	1, =F'9'	12	01	L	1, 40
	BAL	14, SQRT	16	01	BAL	14, 0
	C	1, =F'3'	20	01	C	1, 44
	BNE	ERR	24	01	BC	7, 4
	L	14, SAVE	28	01	L	14, 36
	BR	14	32	0	BCR	15, 14
SAVE	DS	F	34	0	(Skipped for alignment)	
	END		36	00	(Temporary location)	
			40	00	9	
			44	00	3	

Relocating Loader



Relocating Loader

- The four functions of loader were performed by the loader.
  - Allocation bits to solve the problem of relocation
  - Transfer vector to solve the problem of linking
  - Program length to solve the problem of allocation

- Disadvantages
  - Not suited for loading and storing external data.
  - Transfer vector increases the size of object program in memory.
  - Does not provide facility to access data segments that can be shared

6. Loaders

- Function of loader
- General loader scheme
- Absolute loader
- Relocating loader
- Direct linking loader
- Dynamic loading

Direct Linking Loader

- The assembler must give the loader the following information with each procedure or data segment
  - The length of the segment
  - A list of all the symbols in the segment that may be referenced by other segment and their relative location within the segment.
  - A list of symbols not defined in the segment but referenced in the segment.
  - Information as where address constants are located in the segment and a description of how to revise their values.
  - The machine code translation of the source program and the relative addresses assigned.

Program				Translation		
Card No.				**Rel. Loc.**		
1	**JOHN**	**START**				
2		**ENTRY**	**RESULT**			
3		**EXTRN**	**SUM**			
4		**BALR**	**12, 0**	**0**	**BALR**	**12, 0**
5		**USING**	***, 10**			
6		**ST**	**14, SAVE**	**2**	**ST**	**14, 54(0, 12)**
7		**L**	**1, POINTER**	**6**	**L**	**1, 46(0, 12)**
8		**L**	**15, ASUM**	**10**	**L**	**15, 58(0, 12)**
9		**BALR**	**14, 15**	**14**	**BALR**	**14, 15**
10		**ST**	**1, RESULT**	**16**	**ST**	**1, 50(0, 12)**
11		**L**	**14, SAVE**	**20**	**L**	**14, 54(0, 12)**
12		**BR**	**14**	**24**	**BCR**	**15, 14**
				26	**----**	
13	**TABLE**	**DC**	**F'1, 7, 9, 10, 3'**	**28**	**1**	
				32	**7**	
				36	**9**	
				40	**10**	
				44	**3**	
14	**POINTER**	**DC**	**A(TABLE)**	**48**	**28**	
15	**RESULT**	**DS**	**F**	**52**	**----**	
16	**SAVE**	**DS**	**F**	**56**	**----**	
17	**ASUM**	**DC**	**A(SUM)**	**60**	**?**	
18		**END**		**64**		

Direct Linking Loader

- The assembler produce four types of cards
  - ESD (External Symbol Directory)
  - TXT (Text Card)
  - RLD (Relocation and Linkage Directory)
  - END

Relocating Loader

- ESD (External Symbol Directory)
  - This card contain information about
    - all symbols that are defined in this program but that may be referenced elsewhere and
    - all symbols referenced in this program but defined elsewhere
  - 'Type' mnemonic in ESD card can be
    - SD (Segment Definition)
    - LD (Local Definition)
    - ER (External Reference)

Relocating Loader

- ESD (External Symbol Directory)
  - ESD card for example

Reference no.	Symbol	Type	Relative location	Length
1	JOHN	SD	0	64
2	RESULT	LD	52	----
3	SUM	ER	----	----

Direct Linking Loader

- TXT card
  - This card contain actual object code translated version of the source program.

Direct Linking Loader

- TXT card example

Reference number	Relative location	Object code	
4	0	BALR	12, 0
6	2	ST	14, 54(0, 12)
7	6	L	1, 46(0, 12)
8	10	L	15, 58(0, 12)
9	14	BALR	14, 15
10	16	ST	1, 50(0, 12)
11	20	L	14, 54(0, 12)
12	24	BCR	15, 14
13	28	1	
13	32	7	
13	36	9	
13	40	10	
13	44	3	
14	48	28	
17	60	0	

Direct Linking Loader

- RLD card
  - This card contain following information
    - The location of each constant that needs to be changed due to relocation
    - By what it has to be changed
    - The operation to be performed
  - RLD card example

Reference no.	Symbol	Flag	Length	Relative location
14	JOHN	+	4	48
17	SUM	+	4	60

6. Loaders

- Function of loader
- General loader scheme
- Absolute loader
- Relocating loader
- Direct linking loader
- Dynamic loading

Dynamic Loading

- Dynamic loading

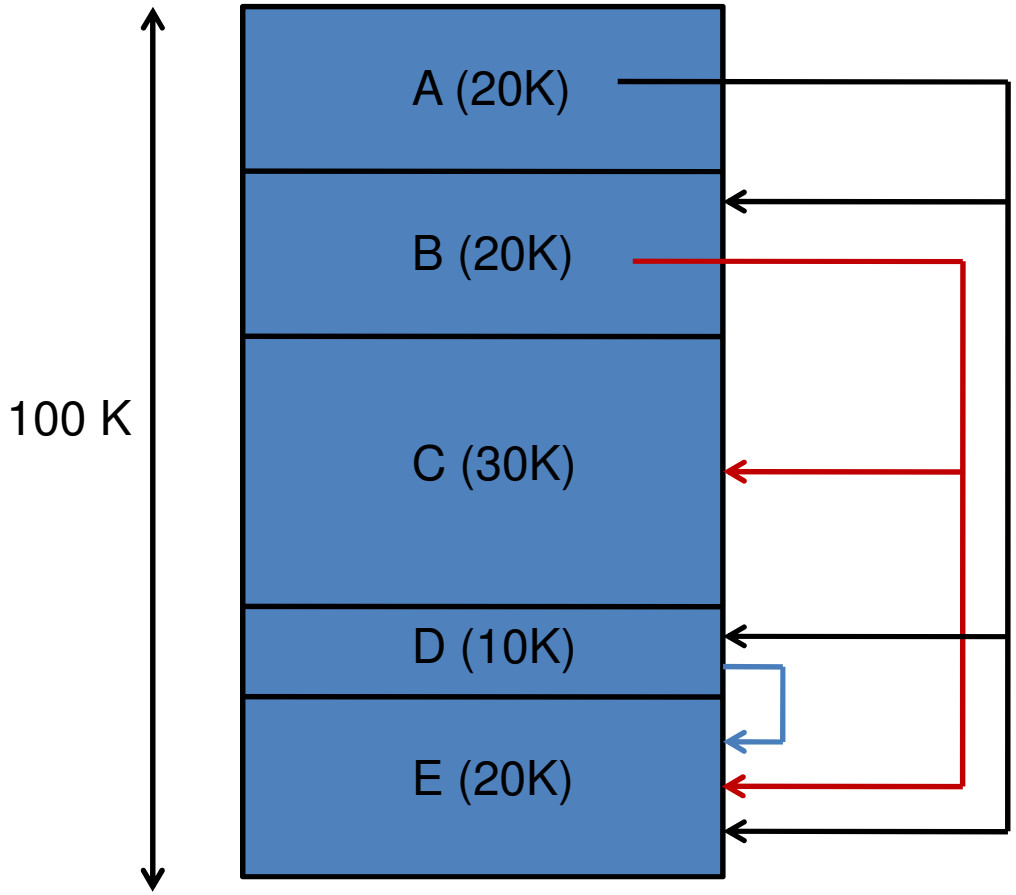


Figure: Subroutine calls between procedure

Dynamic Loading

- Dynamic loading

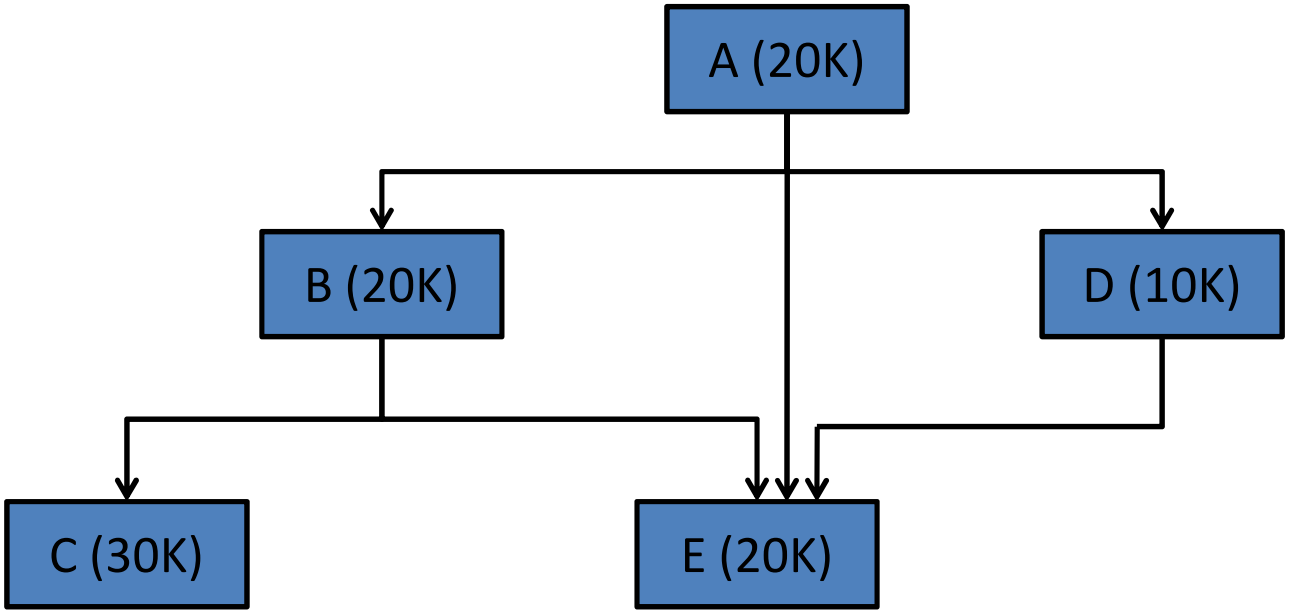


Figure: Overlay structure

Dynamic Loading

- Dynamic loading

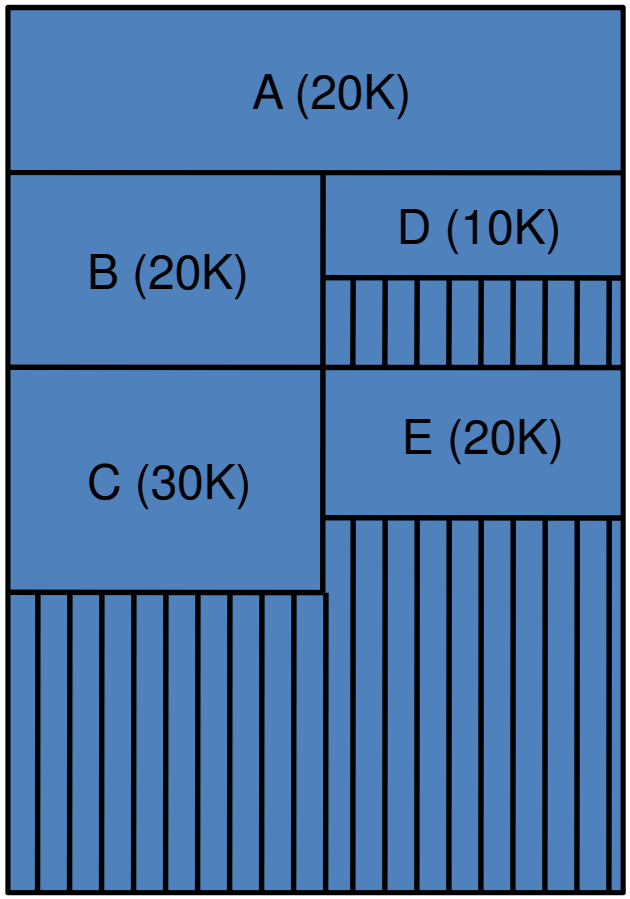


Figure: Possible storage assignment of each procedure