High Performance Computing
Lecture 1

Matthew Jacob

Indian Institute of Science

Agenda

1. Program execution: Compilation, Object files, Function call and return, Address space, Data & its representation (4)
2. Computer organization: Memory, Registers, Instruction set architecture, Instruction processing (6)
3. Virtual memory: Address translation, Paging (4)
4. Operating system: Processes, System calls, Process management (6)
5. Pipelined processors: Structural, data and control hazards, impact on programming (4)
6. Cache memory: Organization, impact on programming (5)
7. Program profiling (2)
8. File systems: Disk management, Name management, Protection (4)
9. Parallel programming: Inter-process communication, Synchronization, Mutual exclusion, Parallel architecture, Programming with message passing using MPI (5)

Reference Material

- Patterson and Hennessy, Computer Organization and Design: The Hardware/Software Interface, 4th Edition ARM Edition, Morgan Kaufman Publishers, 2010.
- Silberschatz, Galvin and Gagne, Operating System Concepts, 8th Edition, John Wiley & Sons, Inc, 2008.
- Bryant and O'Hallaron, Computer Systems: A Programmer's Perspective, Addison Wesley, 2002.

Course Objective

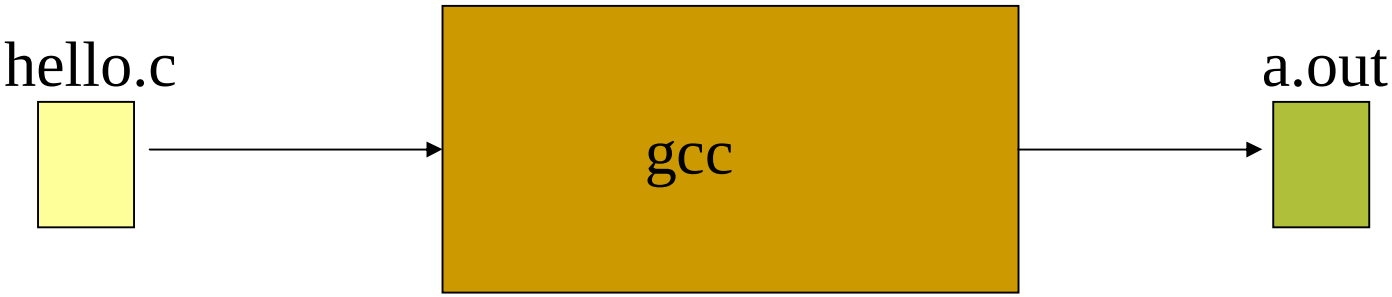
- To understand what happens on a computer system during program execution
- Why?
  - In order to improve the quality of the programs that you write

What is a Computer Program?

- Description of algorithms and data structures to achieve a specific objective
- Could be done in any language, even a natural language like English
- Programming language: A Standard notation for writing programs
- Examples: C, Java, Intel assembly language
- An extreme example: Machine language
- Hence the need for program translators
  - Example: gcc

What does gcc do for you?

% gcc hello.c



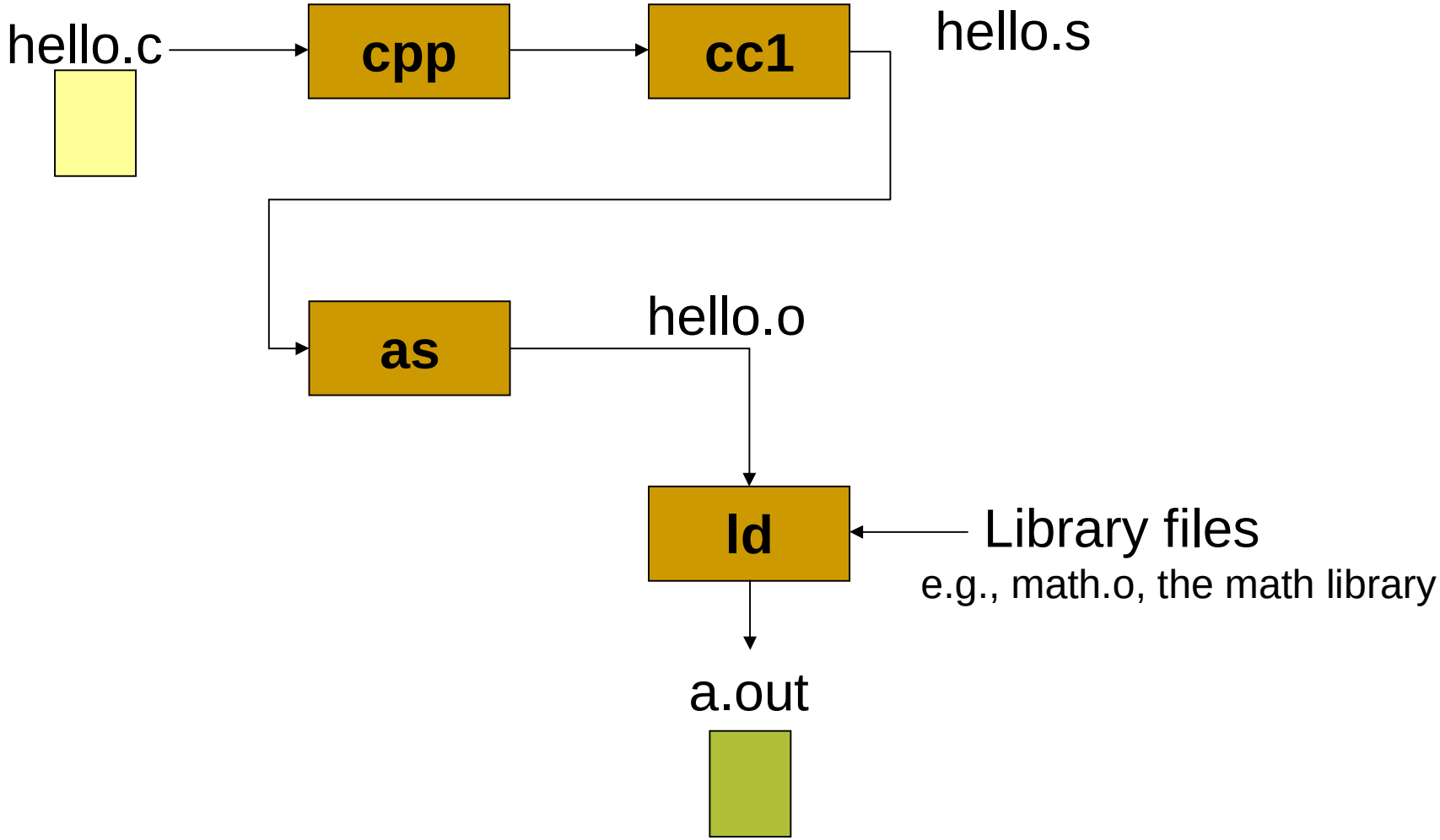
"source": Program that you wrote in the C language and typed into the file hello.c

"object": File generated by gcc. The a.out file contains an equivalent program in machine language that can be executed on a computer system

Steps in gcc

- cpp, cc1, as, ld
- Temporary files are generated in between for use by the next step

Steps in gcc.


hello.c
cpp
cc1
hello.s
as
hello.o
ld
Library files
e.g., math.o, the math library
a.out

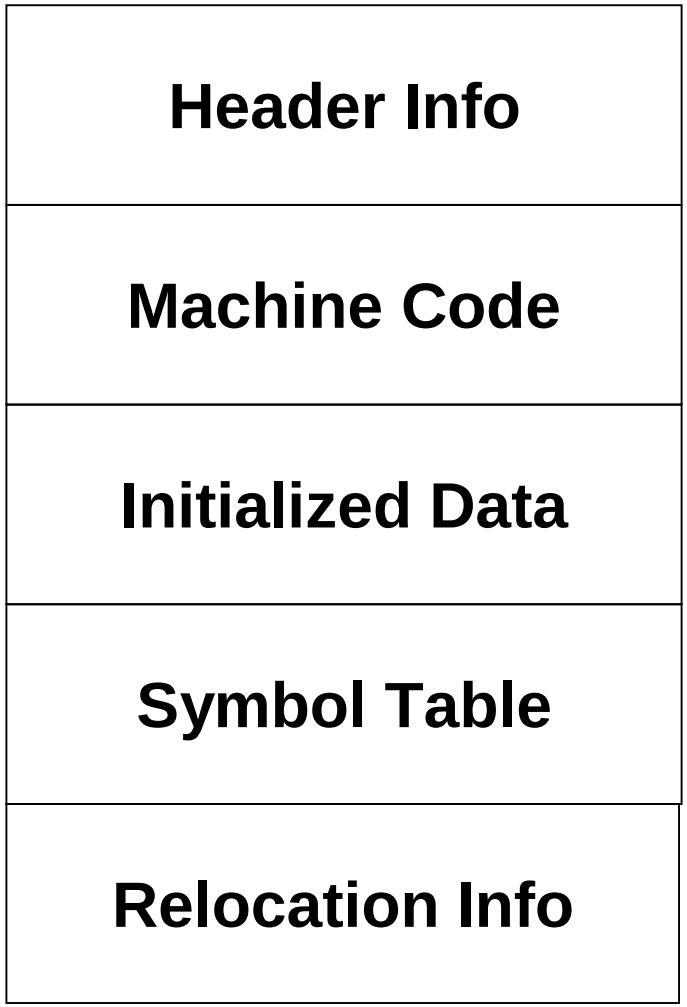
Steps in gcc..

- The a.out file has a well defined format
  - i.e., you can open an a.out file and find specific information or data at specified places
- What does the a.out file contain?
  - Program “code” (machine instructions)
  - Data values (values, size of arrays)
  - Other information that is needed for
    - execution
    - debugging
      - Debugging: A stage in program development where you are identifying the mistakes (“bugs”) in your program

Example

```
#include   <stdio.h>
#include   <math.h>
float arr[100];
int size=100;
void main()
{     int i;
      float sum;
      for  (i=0, sum=0.0; i<size; i++)
      {         arr[i]  =  sqrt(arr[i]);
                sum +=  arr[i];
      }
      printf ("sum is %f\n",sum);
}
```

Object file format



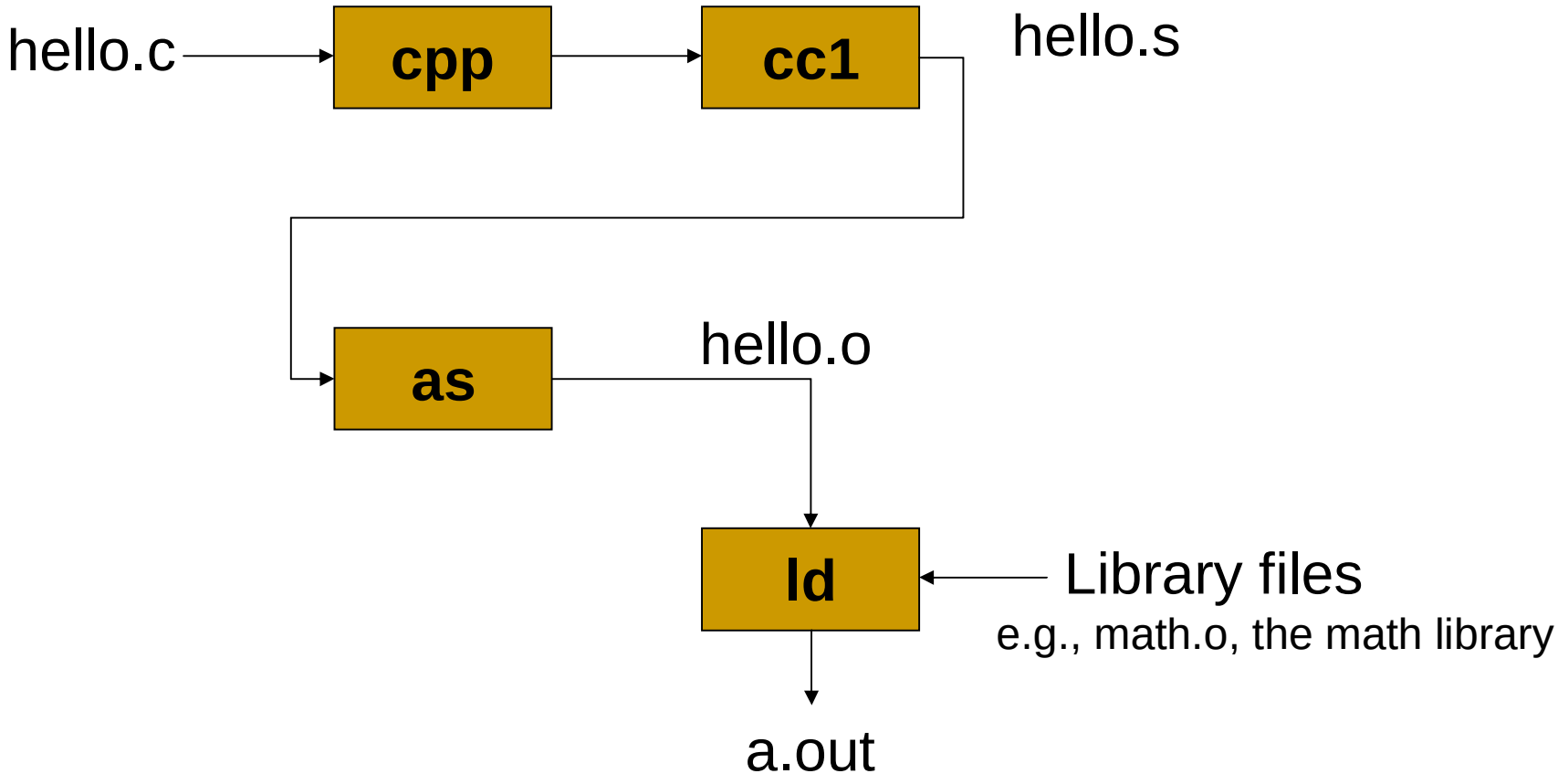
Format of the Object File

Header Info.	0	94	Machine Code Size
	4	4	Initialized Data size
	8	400	Uninitialized Data size
	12	60	Symbol Table
	16	??	Relocation Info
Machine Code	20	??	Start of main; Code for first instruction
	…	…	
	66	??	code for call sqrt

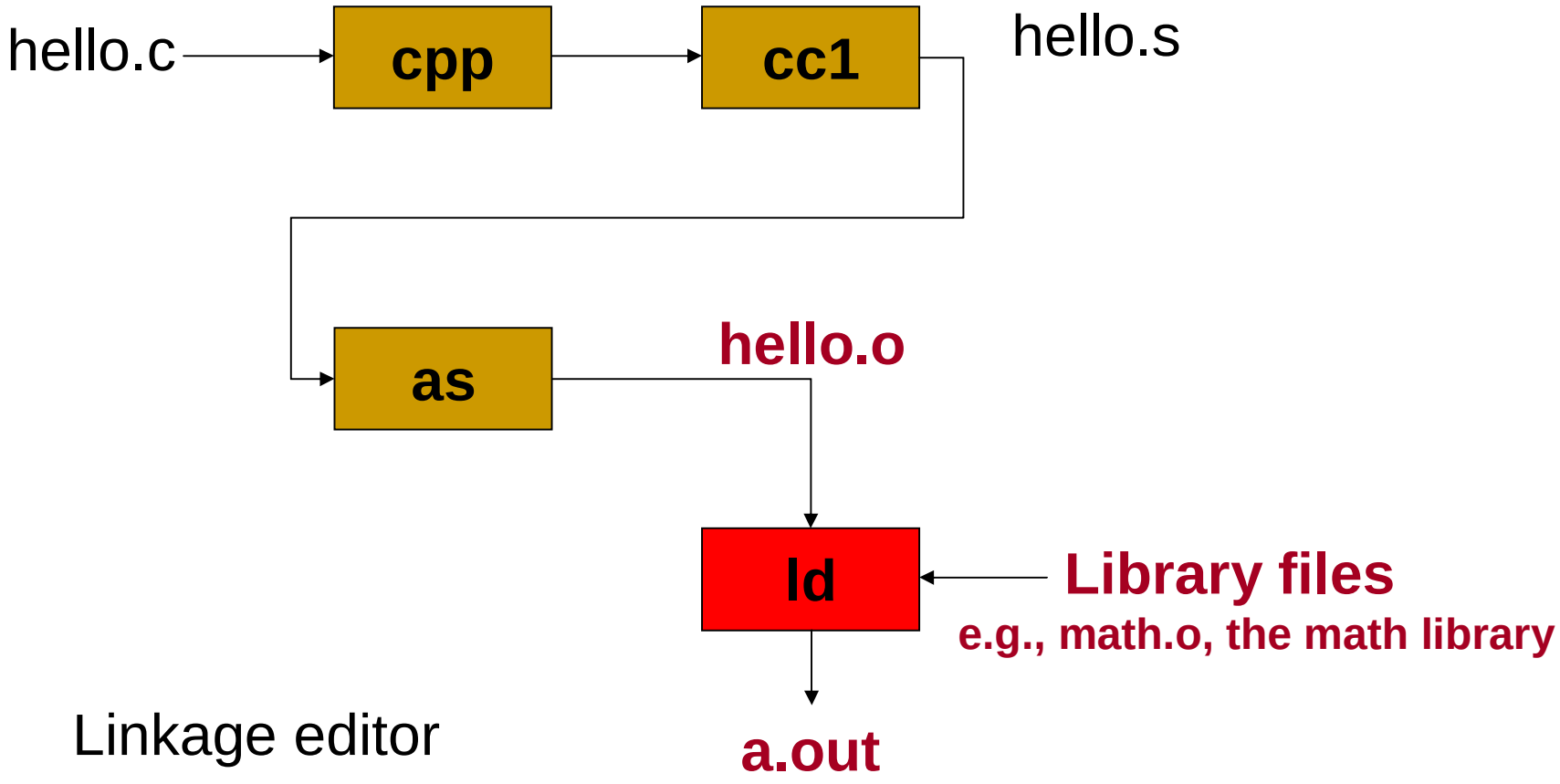
Format of the Object File

Init. Data	114	100	--	Initialized Data
Symbol Table	118	XX	size	Name of symbol “size” and its address
	130	YY	arr	Name of symbol “arr” and its address
	142	ZZ	main	Name of symbol “main” & its address
	154	??	Sqrt	Name of symbol “sqrt” and its address
	166	??	printf	Name of symbol “printf” & its address
Relocation Info	178	??	Info. on offsets at which external variables are called	

Steps in gcc.


hello.c
cpp
cc1
hello.s
as
hello.o
ld
Library files
e.g., math.o, the math library
a.out

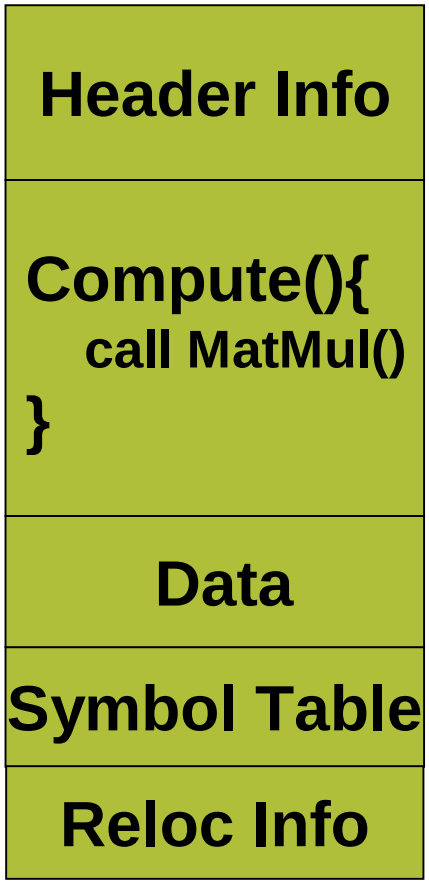
Steps in gcc.

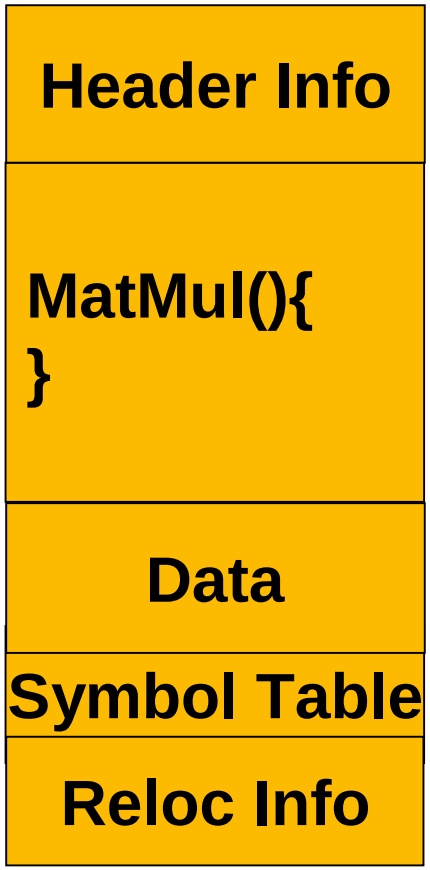


ld Linkage editor

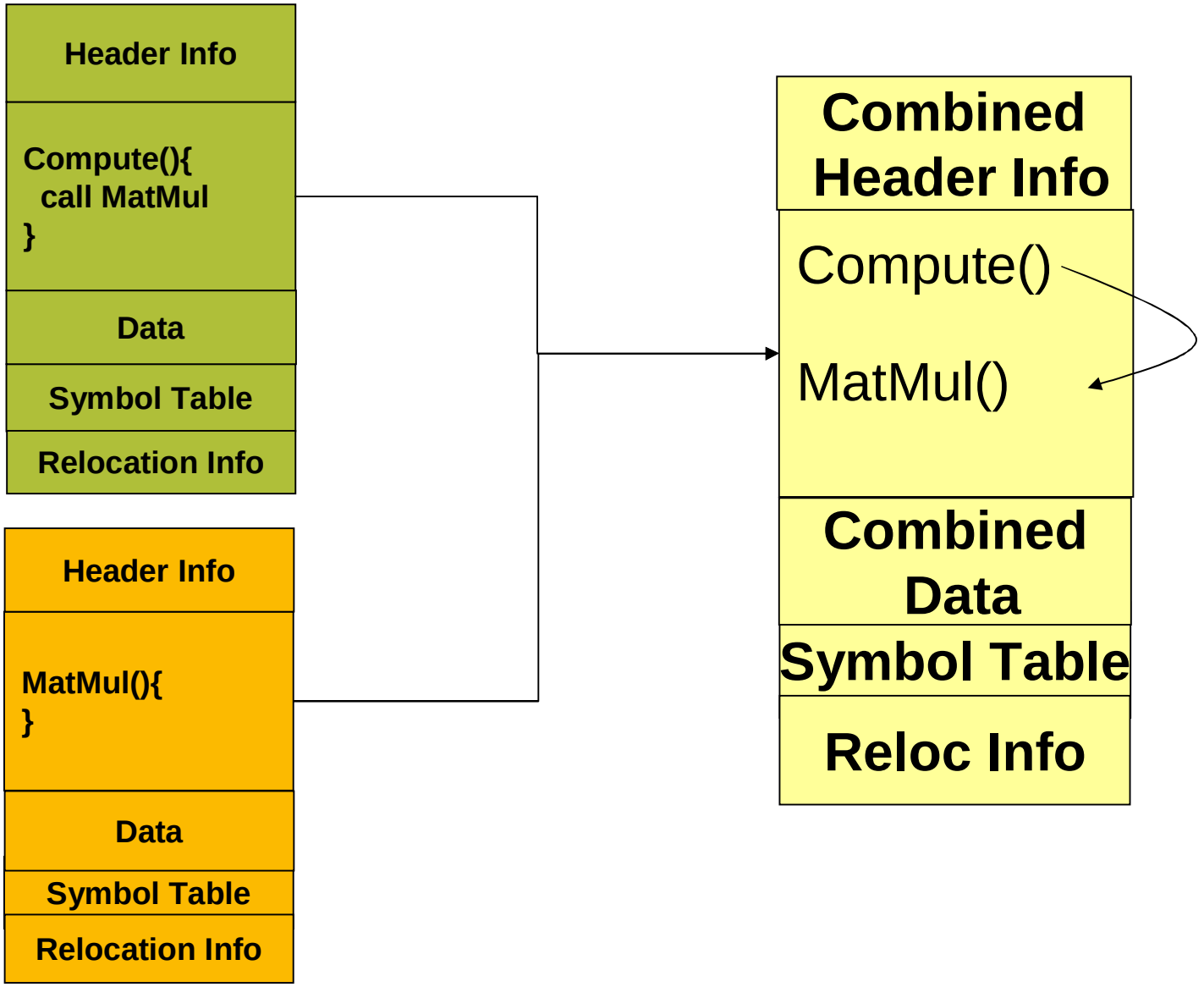
Merges 2 or more object files into one

Linking Multiple Modules





Linking Multiple Modules



Program = Instructions + Data

- We will next learn more about data

There are different kinds of data

How does one piece of data differ from another?

- Constant vs variable
- Basic vs structured
- Of different types
  - Character
  - Integer (unsigned, signed)
  - Real
  - Others (boolean, complex, …)

Data differing in their lifetimes

- Lifetime: Interval between time of creation and end of existence
- How long can the lifetime of a datum be?
- We will consider 3 possible lifetimes

Program Data: Different Lifetimes.

1. Lifetime = Execution time of program
   - Initialized/uninitialized data
   - Must be indicated in executable file
   - The space for all of this data can be assigned when program execution starts (Static Allocation)

Program Data: Different Lifetimes.

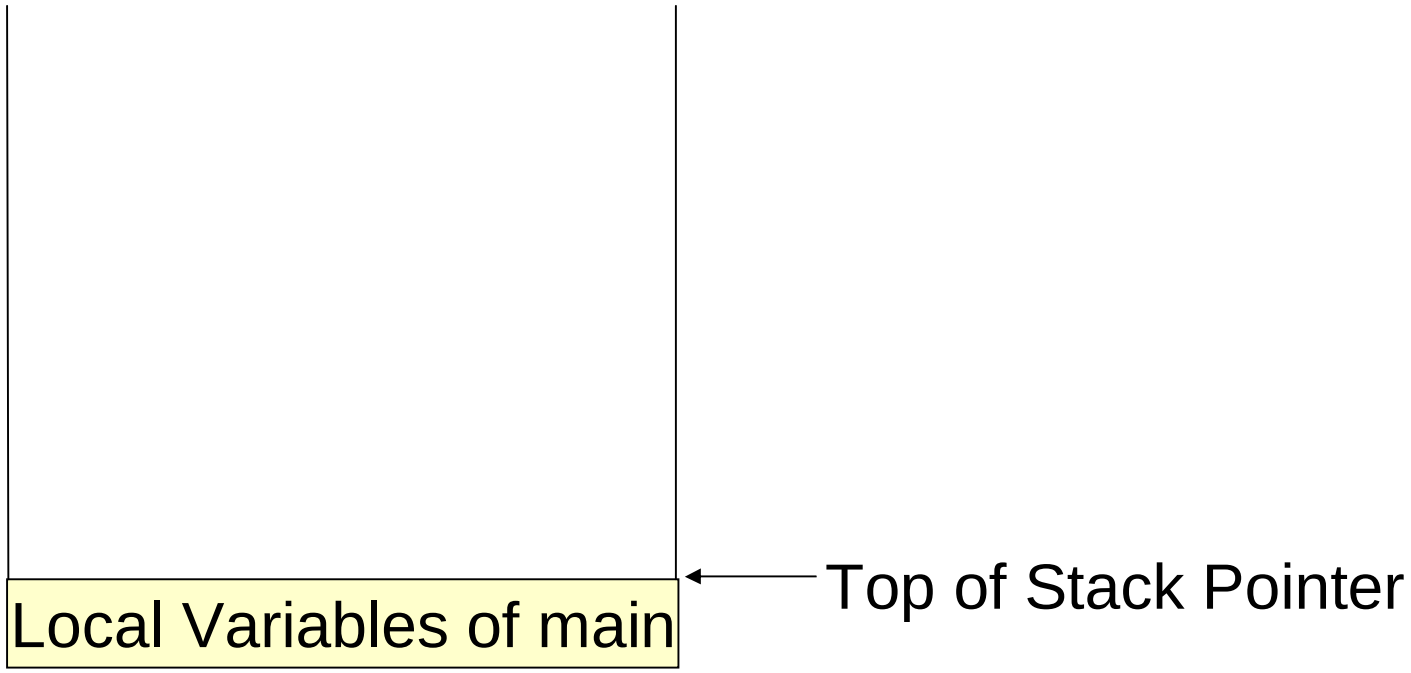
1. Lifetime = Execution time of program
2. Lifetime = Time between explicit creation of data & explicit deletion of data
   - Dynamic memory allocation
   - In C you create new data using a function like malloc()
   - The space for this data is managed dynamically when the malloc/free is executed (Heap allocation)

Program Data: Different Lifetimes..

1. Lifetime = Execution time of program
2. Lifetime = Time between explicit creation of data & explicit deletion of data
3. Lifetime = During execution of a function (i.e., time between function call and return)
   - Local variables, parameters of the function
   - The space for this data is assigned when the function is called and reclaimed on return from the function (Stack allocation)
   - Stack: Like a pile of books on a table

Stack allocated: Function Local Variables

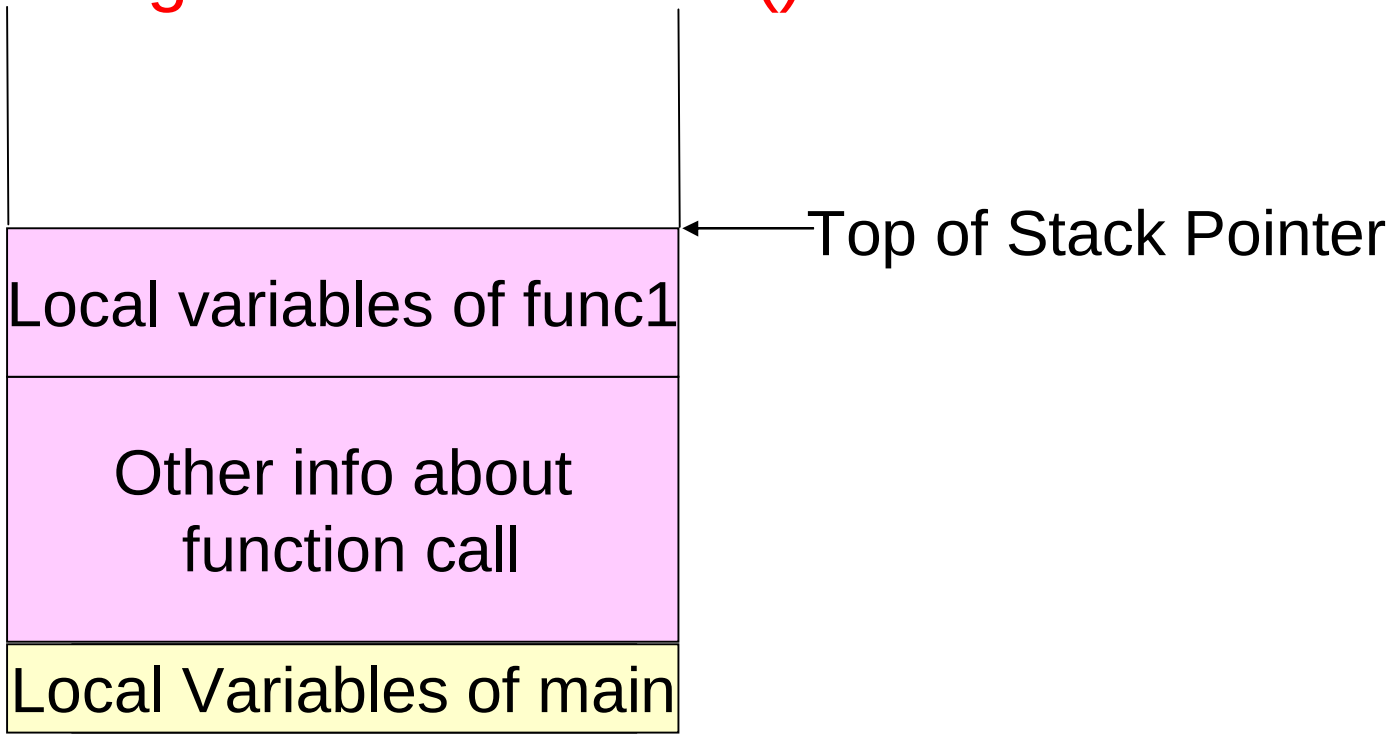
When the program starts executing



What if main() then calls function func1()?

Stack allocated: Function Local Variables.

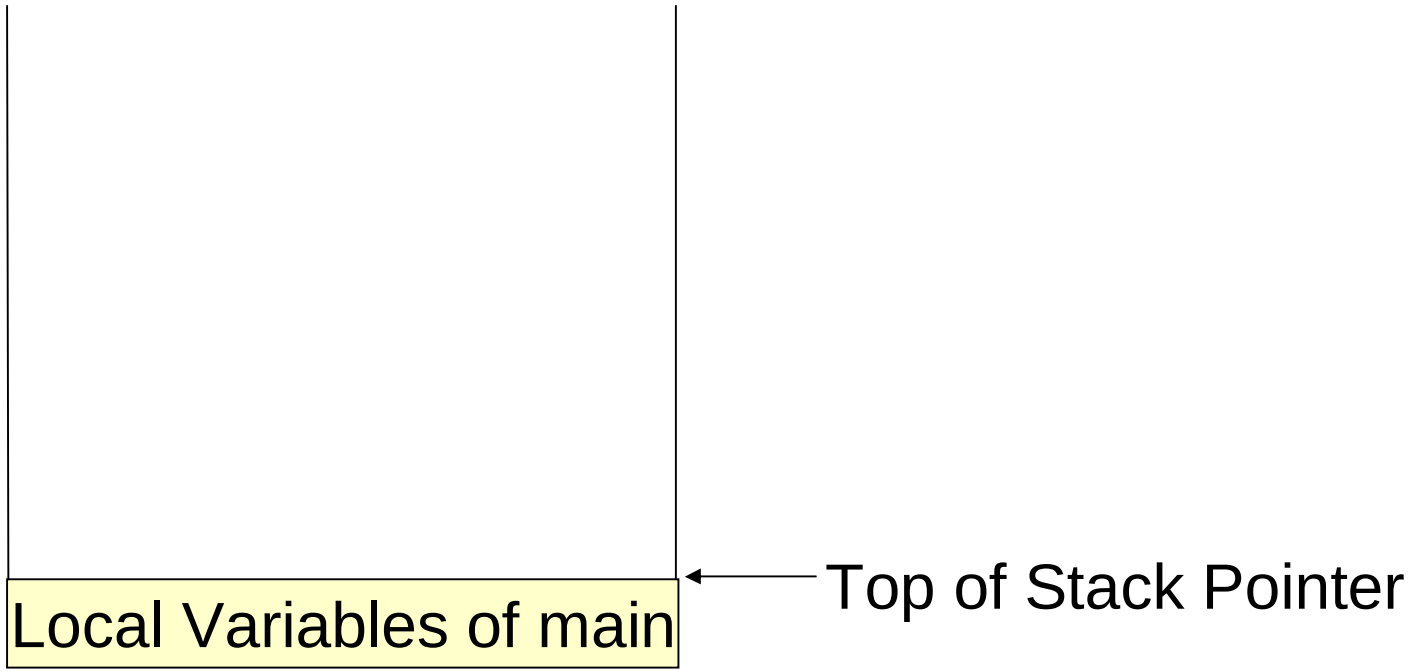
While executing in function func1()



What happens on return from the call to func1()?

Stack allocated: Function Local Variables..

Executing in main() once again



During program execution

Code (machine language program)

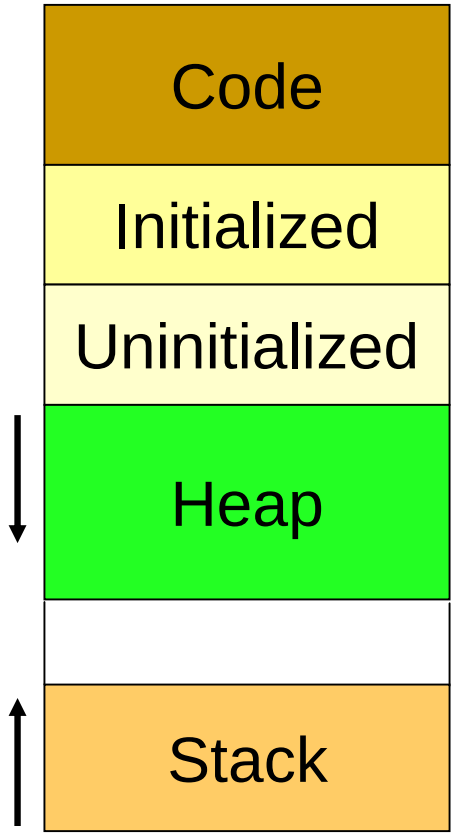
Data (initialized and uninitialized)

Code and Data don't change in size while the program is executing

Heap (for dynamically allocated data)

Stack (for function local variables)

Heap and Stack change in size as program executes



How is Data Represented?

- On a digital computer
- Binary
  - Base 2 number system
  - Two values: 0 and 1
  - Bit (Notation: b); Byte (Notation: B) 8 bits
  - Other notation: K, M, G, T, P etc
    - K: 2^{10} , M: 2^{20} , G: 2^{30}, etc
    - “2 GB of RAM”, “1 TB hard disk drive”

Character Data

- Typically represented using the ASCII code
- ASCII: American Standard Code for Information Interchange
- Each character is represented by a unique 8 bit ASCII code word
- Example: ‘a’ is represented by 01100001, ‘1’ is represented by 00110001

How is Data Represented?

- Character data: ASCII code
- Integer data