Introduction to Embedded Systems - Lesson 4: Embedded Software in the System

By Prof. Amiya Kumar Rath

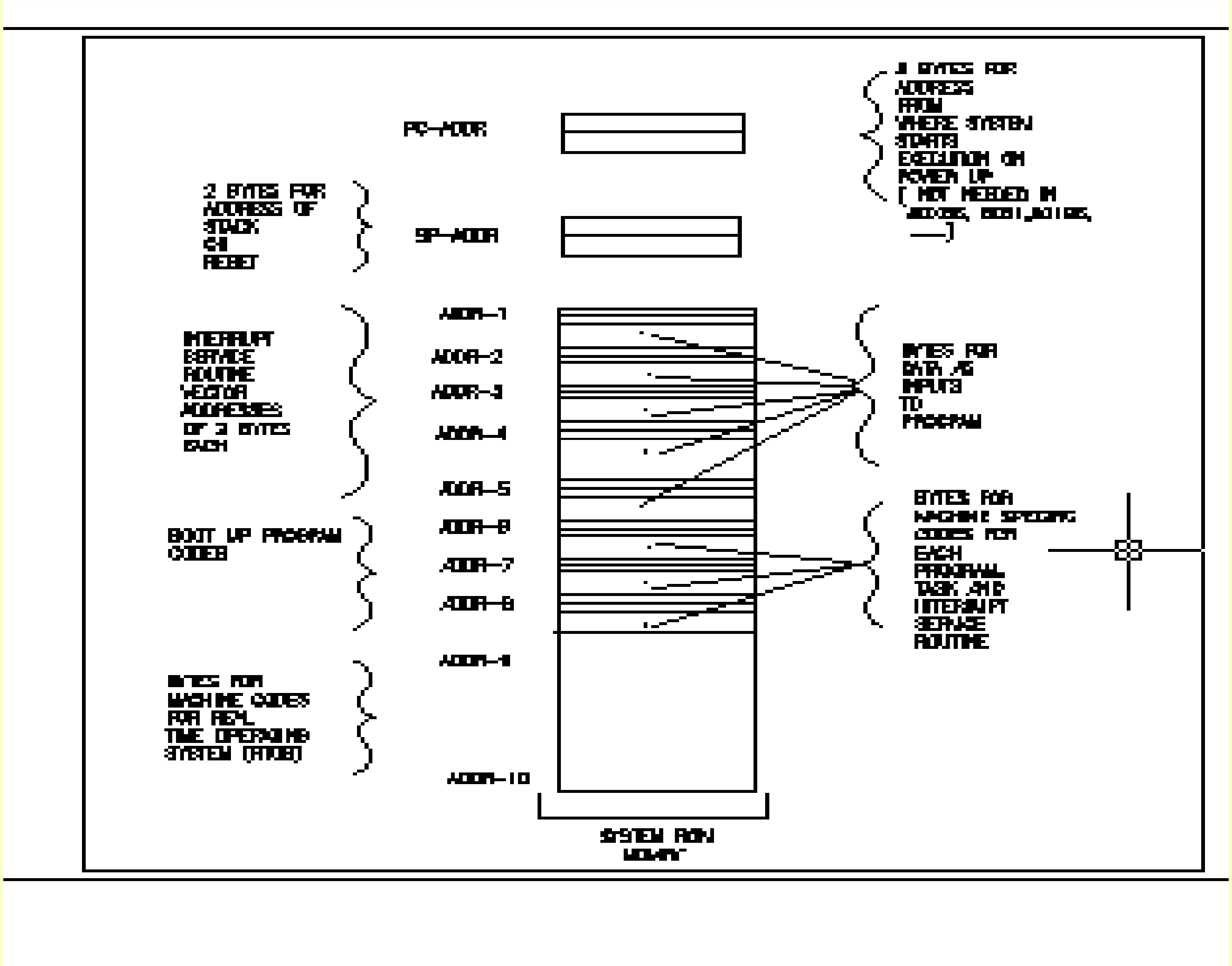
1. ROM Image

Final stage software is also called ROM image. Why? Just as an image is a unique sequence and arrangement of pixels, embedded software is also a unique placement and arrangement of bytes for instructions and data.

Final machine codes of software

The final machine codes of software for a product embeds in the ROM (or PROM) as an image at a frame. Bytes at each address must be defined for creating the ROM image. By changing this image, the same hardware platform will work differently and can be used for entirely different applications or for new upgrades of the same system.

Example of ROM Addresses


3 BYTES FOR ADDRESS FROM WHERE SYSTEM STARTS EXECUTION ON POWER UP [NOT NEEDED IN 8086, 8051, 80196, ——]
PC-ADDR
2 BYTES FOR ADDRESS OF STACK ON RESET
SP-ADDR
ADDR-1
ADDR-2
ADDR-3
ADDR-4
ADDR-5
ADDR-6
ADDR-7
ADDR-8
ADDR-9
ADDR-10
INTERRUPT SERVICE ROUTINE VECTOR ADDRESSES OF 3 BYTES EACH
BYTES FOR DATA AS INPUTS TO PROGRAM
BOOT UP PROGRAM CODES
BYTES FOR MACHINE SPECIFIC CODES FOR EACH PROGRAM TASK AND INTERRUPT SERVICE ROUTINE
BYTES FOR MACHINE CODES FOR REAL TIME OPERATING SYSTEM (RTOS)
SYSTEM ROM MEMORY

Distinct ROM image in a distinct Embedded System

Hardware elements between the distinct systems can be identical but it is the software that makes a system unique and distinct from the other. Therefore, a ROM image is also distinct for a distinct embedded system product

Compressing the Codes and Data

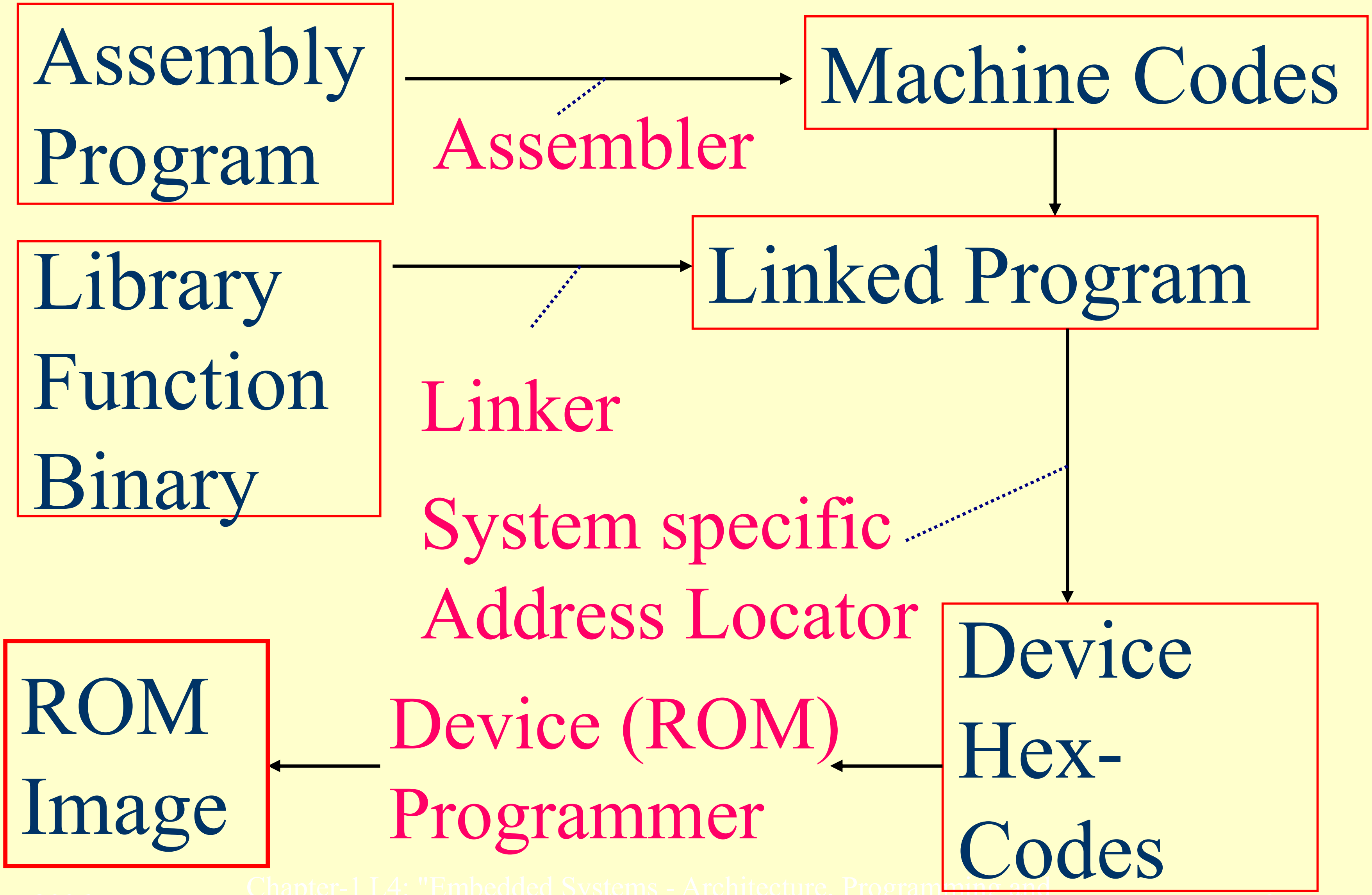
- ROM image may alternatively have the software (for example, the zip format) and data (for example, the pictures in jpg or gif format) along with the required software for decompression at run-time.

2. Assembly Language Codes

Needed for Invoking Processor Specific Instructions

For configuring some specific physical device or subsystem like transceiver, the machine codes can be straight away used. For physical device driver codes or codes that utilize the processor-specific features-invoking codes, 'processor-specific' assembly language is used. A file is then created in three steps using 'Assembler', 'Linker' and 'Locator' before finally burned at the ROM

Assembly to ROM-Image Conversion



Assembly Program

Machine Codes

Assembler

Library Function Binary

Linked Program

Linker

System specific Address Locator

Device Hex-Codes

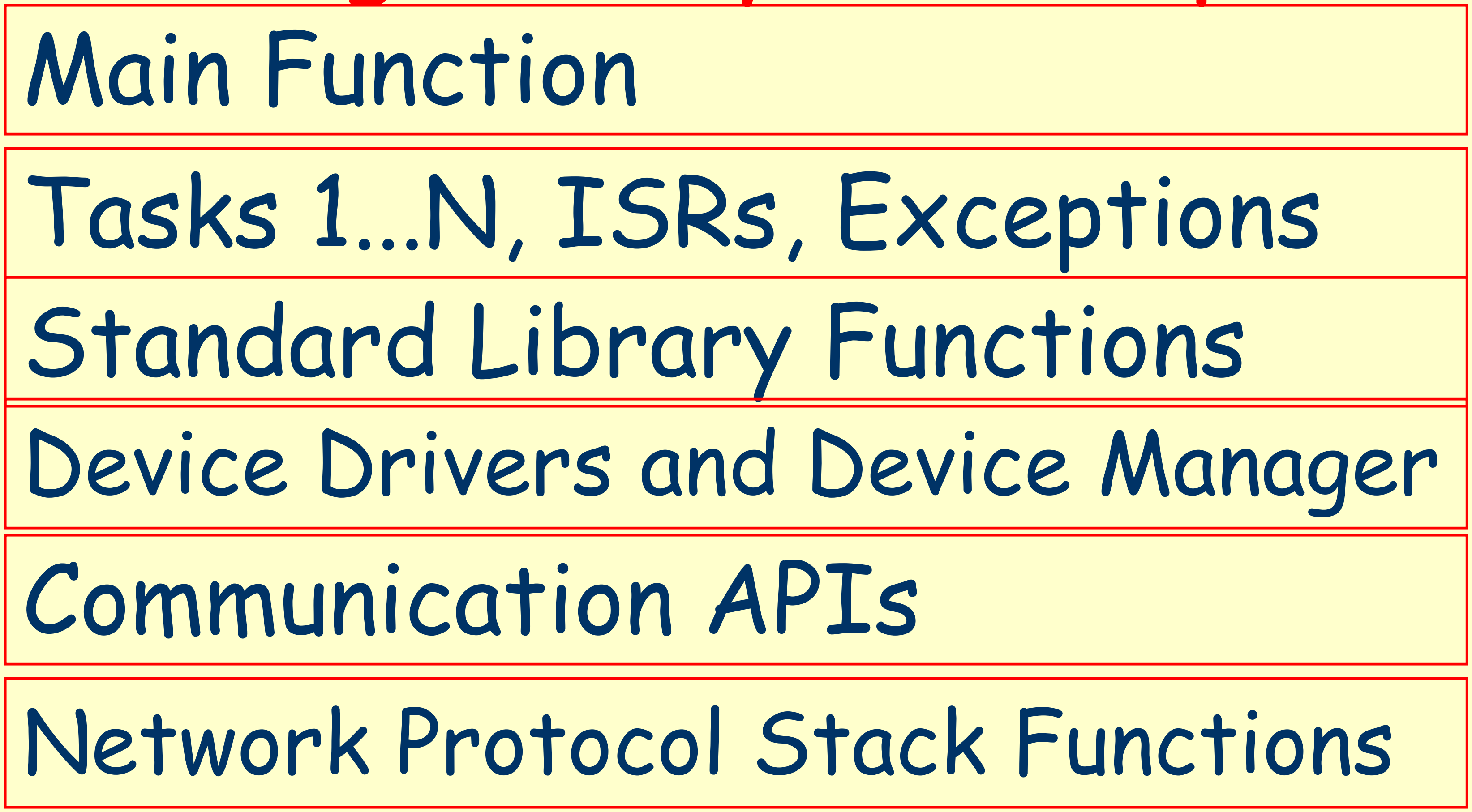
Device (ROM) Programmer

ROM Image

3. Application Software - Different Program Layers

- For most systems, software codes are developed in C. C program has various layers; processor commands, main function, task functions and library functions, interrupt service routines and kernel (scheduler). The compiler generates an object file. Using linker and locator, the file for ROM image is created for the targeted hardware. C++ and Java are object oriented languages used for coding.

C Program Layer Examples



Main Function

Tasks 1…N, ISRs, Exceptions

Standard Library Functions

Device Drivers and Device Manager

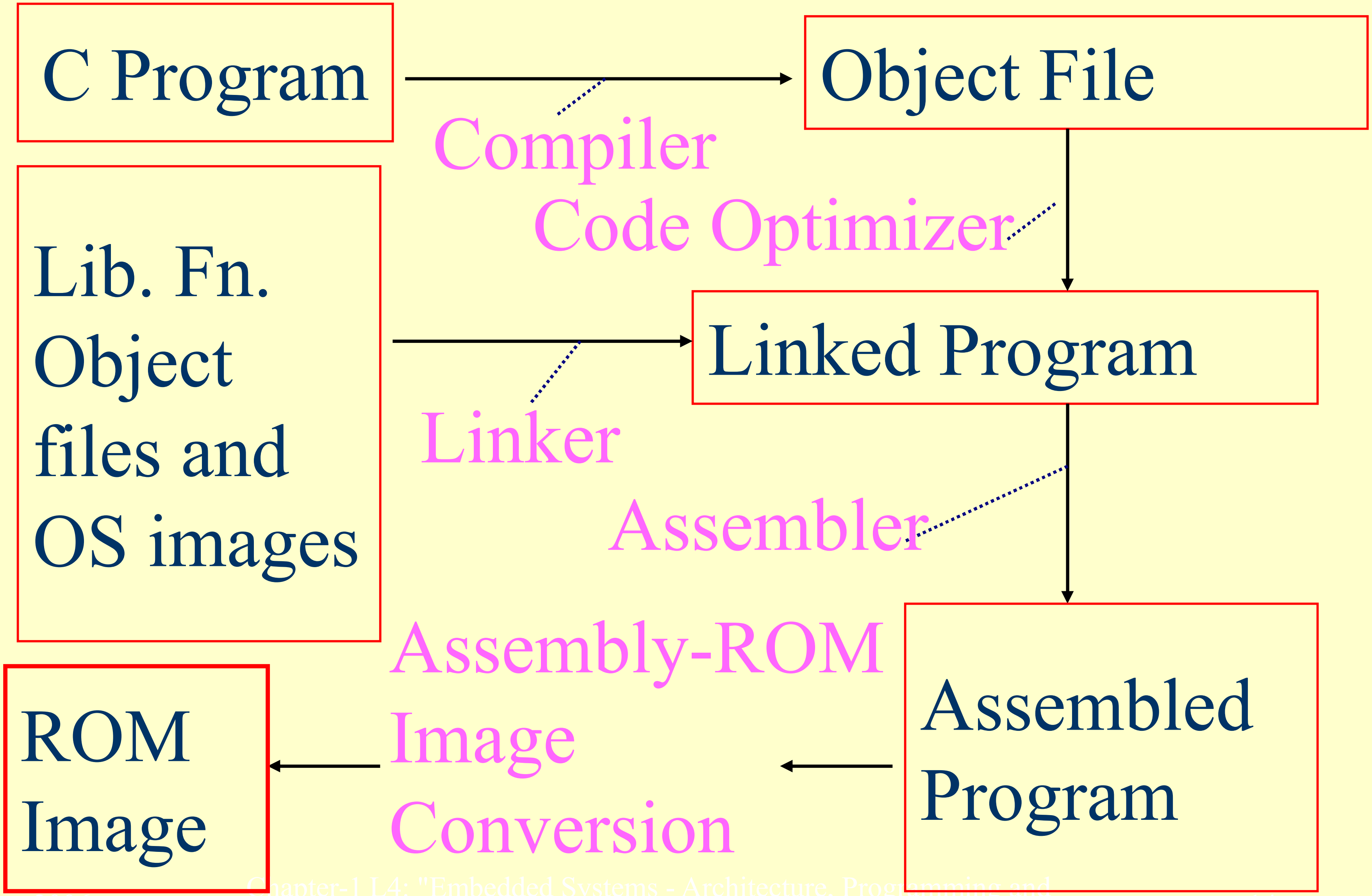
Communication APIs

Network Protocol Stack Functions

RTOS Program Layers Example

Embedded operating system images, based on the hardware and software requirements of the embedded design

C program to Assembly Conversion



C Program

Compiler

Object File

Code Optimizer

Lib. Fn. Object files and OS images

Linker

Linked Program

Assembler

Assembled Program

Assembly-ROM Image Conversion

ROM Image

4. Application Software Development Tools

Platform Builder (Application Developer) or integrated development environment (IDE).

RTOS for example, Windows CE or VxWorks

RTOS is needed when the tasks for the system have real time constraints and deadlines for finishing the tasks

Real Time Operating System (RTOS)

Embedded software is most often designed for performing multiple actions and controlling multiple devices and their ISRs. Multitasking software is therefore essential. For scheduling multiple tasks, popular, readily available RTOS kernel with functions (like device-drivers) are most often used.

Definition of Real Time System

- A System that is composed of two or more concurrent processes that execute with stringent timing requirements and cooperate with each other to accomplish a common goal.

Important RTOSs

- μCOS-II
- VxWorks
- QNX

Important RTOSs

- RTLinux
- Windows CE

5. Devices in a System

- In an embedded system, there are a number of *physical devices*. Exemplary physical devices- Mobile Device has keypad, LCD display unit, Modem radio interface, port and network drivers.

Device Drivers

A **device driver** is software for controlling, receiving and sending a byte or a stream of bytes from or to a device.

Driver controls three functions :

1. Placing appropriate bits at the control register or word.
2. Calling an ISR on interrupt (event) or on setting a status flag in the status register, and run (drive) the ISR (also called Interrupt Handler Routine).
3. Resetting the status flag after interrupt service.

Device Functions Manager

- *Device Management* software (usually a part of the OS) provide codes for detecting the presence of devices, for initializing these, memory buffers allocation, and testing the active devices

- The driver modules may also include software for allocating and registering port (in fact, it may be a register or memory) addresses for the various devices at distinctly different addresses, including codes for detecting any collision between these, if any

6. Development Tools

1. Editor,
2. Interpreter,
3. Compiler,
4. Assembler and Cross Assembler, IDE,
5. Prototyper

Project Manager Tool

To manage the files that associates with a design stage project and keep several versions of the source file(s) in an orderly fashion.

Source Code Engineering Tools

- Simulator… To simulate the target processor and hardware elements on a host PC and to run and test the executable module

Source Code Engineering Tools

- Debugging and Testing Tools, including tools for Navigation, Profiler, Stethoscope, Trace Scope and others
- Table 1.6 for details

Summary

We learnt:

- (i) embedded software saves as ROM image
- (ii) program layers in the embedded software
- (iii) conversion steps from high level language codes to assembly and then to the ROM image

We learnt:

The main elements of an embedded software

The application software, RTOS, device drivers, device function manager and final software saves as ROM image

We learnt:

- high level language used for a software development is mostly C, C++ or Java

End of the Lesson -4