
CSPC61, EMBEDDED SYSTEMS AND ARCHITECTURE
CHAPTER-10: MIDDLEWARE AND APPLICATION SOFTWARE

---

1. What is middleware?

Middleware software is any system software that is not the OS kernel, device drivers, or application software. In short, in an embedded system middleware is system software that typically sits on either the device drivers or on top of the OS and can sometimes be incorporated within the OS itself. It is an abstraction layer generally used on embedded devices with two or more applications to provide flexibility, security, portability, connectivity, intercommunication, and/or interoperability mechanisms between applications.

---

2. Which of Figures 10-36a, b, c, and d is incorrect in terms of middleware software into the Embedded Systems Model? **Example1**

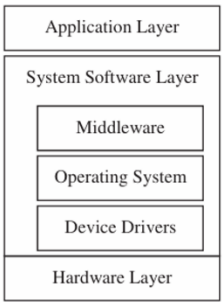

Figure 10-36c: Example 3

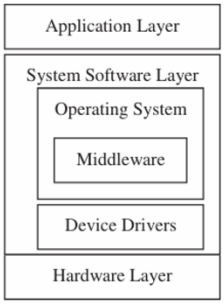

Figure 10-36d: Example 4

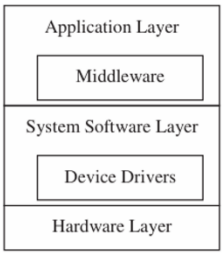

Figure10-36a: Example 1

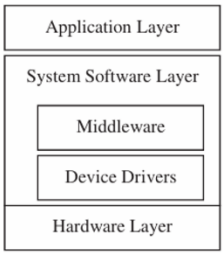

Figure 10-36b: Example 2

---

3. What is the difference between general-purpose middleware and market-specific middleware? List two real-world examples of each.

Parameter	*General-Purpose*	*Market-Specific*
Definition	Typically implemented across various devices and applications.	Unique to specific families of embedded systems or industries.
Examples	Networking protocols (e.g., TCP/IP), file systems, virtual machines (e.g., JVM)	Digital TV standard-based software like DVB or ASTC, automotive communication protocols like CAN bus

---

4. Where in the OSI model is networking middleware located?

Networking middleware typically operates above the device driver layer and below the application layer of the OSI model. Therefore, it is situated in the transport layer (Layer 4) and the session layer (Layer 5) of the OSI model.

---

5. Draw the TCP/IP model layers relative to the OSI model. Which layer would TCP fall under?

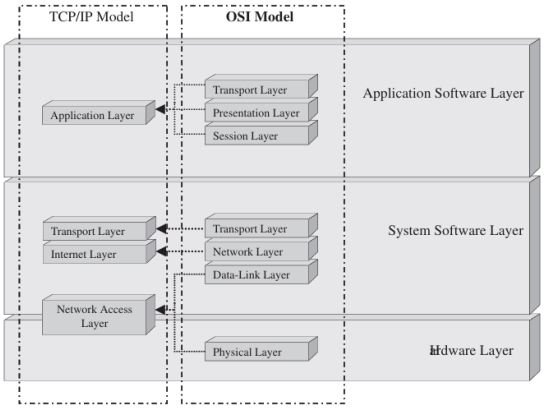

Figure 10-4a: TCP/IP, OSI Models and Embedded Systems Model block diagram

---

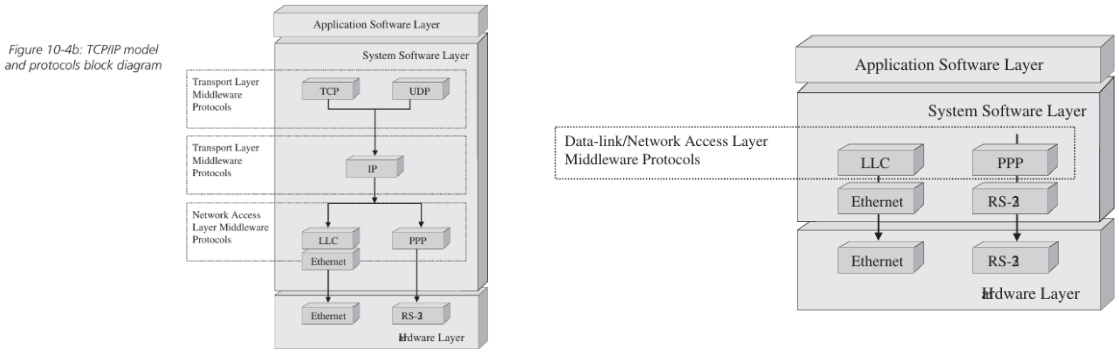

Figure 10-4b: TCP/IP model and protocols block diagram

TCP falls under the transport layer.

---

6. RS-232 related software is middleware. True or False? **True**

---

7. PPP manages data as: (Options: A) frames / B) datagrams / C) messages / D) All of the above / E) None of the above) – **A) frames**

---

8. Name and describe the four subcomponents that make up PPP software. What RFCs are associated with each?

- The *PPP encapsulation mechanism* (in RFC 1661) such as the high-level data-link control (HDLC) framing in RFC1662 or the link control protocol (LCP) framing defined in RFC 1661 to process (i.e., demultiplex, create, verify checksum, etc).
- *Data-link protocol handshaking*, such as the link control protocol (LCP) handshaking defined in RFC 1661, responsible for establishing, configuring, and testing the data link connection.
- *Authentication protocols*, such as PAP (PPP authentication protocol) in RFC 1334, used to manage security after the PPP link is established.
- *Network control protocols (NCP*), such as IPCP (Internet protocol control protocol) in RFC 1332, that establish and configure upper-layer protocol (i.e., OP, IPX, etc.) settings.

---

9. What is the difference between a PPP state and a PPP event? List and describe three examples of each.

At any given time, a PPP connection on a device is in a particular *state*. On the other hand, *events* are what cause a PPP connection to transition from state to state.

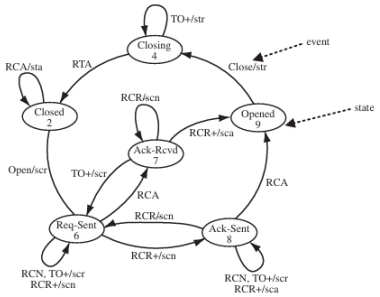

Figure 10-9: PPP connection states and events [10-1]

---

10. What is an IP address? What networking protocol processes IP addresses?

The source and destination IP address fields in an IP Datagram packet are the networking addresses, also commonly referred to as the Internet or IP address, processed by the IP layer. IP addresses are 32 bits long, in “dotted-decimal notation,” divided by “dots” into four octets (four 8-bit decimal numbers between the ranges of 0-255 for a total of 32 bits).

The networking layer protocol called Internet Protocol, or IP, processes IP addresses.

---

11. What is the main difference between UDP and TCP?

The main difference between UDP (User Datagram Protocol) and TCP (Transmission Control Protocol) lies in their connection-oriented nature. TCP provides a reliable, connection-oriented communication, ensuring data delivery and ordering, while UDP is connectionless and does not guarantee delivery or ordering of packets.

---

12. Name three embedded JVM standards that can be implemented in middleware. What are the differences between the APIs of these standards? List two real-world JVMs that support each of the standards.

Three embedded JVM standards that can be implemented in middleware are: Personal Java (pJava), Java 2 Micro Edition (J2ME) and Connected Device Configuration (CDC)

The differences between the APIs of these standards lie in the scope and functionality provided by each standard's Java APIs. For example: PJava offers APIs suitable for personal and embedded devices with a focus on simplicity and resource efficiency. J2ME provides APIs tailored for small, resource-constrained devices with support for mobile and embedded applications. CDC extends the capabilities of J2ME by offering a more comprehensive set of APIs suitable for connected devices, including networking, security, and advanced media capabilities.

Real-world JVMs that support each of these standards: Personal Java (pJava): Oracle's PersonalJava and IBM's PersonalJava implementation for embedded systems. Java 2 Micro Edition (J2ME): Oracle's Java ME SDK and BlackBerry's Java ME runtime environment. Connected Device Configuration (CDC): Oracle's Java Embedded Suite and IBM's WebSphere Micro Environment (WME).

---

13. [T/F] The .NET compact framework is implemented in the middleware layer of the Embedded Systems Model.

False. The .NET Compact Framework is not typically implemented in the middleware layer of the Embedded Systems Model. Instead, it is primarily utilized as an application development framework for embedded systems, providing a runtime environment and libraries for developing and running .NET applications on resource-constrained devices.

---

14. What is application software? Where in the Embedded Systems Model is application software typically located?

Application software refers to computer programs or software applications designed to perform specific tasks or functions for end-users. In the Embedded Systems Model, application software is typically located in the upper layers, specifically in the application layer.

---

15. Name two examples of application-layer protocols that can either be implemented as stand-alone applications whose sole function is that protocol, or implemented as a sub-component of a larger multi-function application.

HTTP (Hyper-Text-Transfer-Protocol), SMTP (Simple-Mail-Transfer-Protocol) and FTP (File-Transfer Protocol) are examples.

---

16. What is the difference between an FTP client and an FTP server? What type of embedded devices would implement each?

The main difference between an FTP client and an FTP server lies in their roles and functionalities within the FTP protocol. An FTP client is responsible for initiating connections to FTP servers, transmitting commands to the server, and requesting file transfers. On the other hand, an FTP server is responsible for listening for incoming connections from FTP clients, interpreting commands received from clients, and facilitating file transfers.

Embedded devices that typically implement an FTP client include: MP3 players and Embedded systems with file transfer capabilities, such as IoT devices or networked appliances.

Embedded devices that typically implement an FTP server include: Network-attached storage (NAS) devices and Embedded systems acting as servers for remote file access or updates, such as industrial control systems or embedded servers in smart home devices.

---

17. SMTP is a protocol typically implemented in A) an e-mail application / B) a kernel / C) a BSP / D) every application / E) None of the above. Answer: **A. an e-mail application**.

---

18. SMTP typically relies on TCP middleware to function. True or False? **True**

---

19. What is HTTP? What types of applications would incorporate an HTTP client or server?

HTTP (Hypertext Transfer Protocol) is an application layer protocol widely used to transmit various types of data over the Internet. It operates on a client-server model and relies on TCP as its underlying transport protocol.

Applications that would incorporate an HTTP client or server include:

- Web browsers (incorporating an HTTP client to request web pages)
- Web servers (incorporating an HTTP server to respond to client requests for web pages)

---

20. What type of programming languages would introduce a component at the application layer?

Programming languages used for developing applications typically introduce components at the application layer. These languages include High-level programming languages such as: Java, Python, C#, JavaScript, Ruby, Swift, Kotlin, Go and PHP. These languages provide developers with the tools and frameworks necessary to build various types of applications, including web applications, mobile apps, desktop software, and more, all of which operate at the application layer of the OSI model.

CSPC61, EMBEDDED SYSTEMS AND ARCHITECTURE
CHAPTER-11: DEFINING THE SYSTEM – CREATING THE ARCHITECTURE AND DOCUMENTING THE DESIGN

---

1. Draw & describe the 4 phases of the Embedded System Design and Development Lifecycle Model.

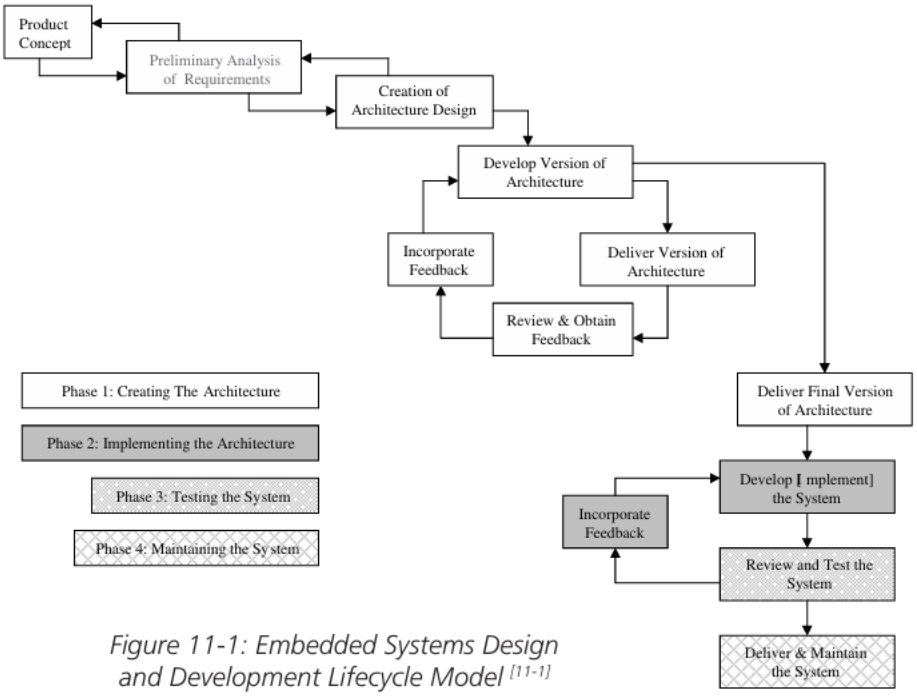

Figure 11-1: Embedded Systems Design and Development Lifecycle Model [11-1]

This model indicates that the process of designing an embedded system and taking that design to market has four phases:

- *Phase-1*: Creating the Architecture, which is the process of planning the design of the embedded system.
- *Phase-2*: Implementing the Architecture, which is the process of developing the embedded system.
- *Phase-3*: Testing the System, which is the process of testing the embedded system for problems, and then solving those problems.
- *Phase-4*: Maintaining the System, which is the process of deploying the embedded system into the field and providing technical support for users of that device for the duration of the device's lifetime.

---

2. Of the four phases, which phase is considered the most difficult and important? Why?

The most important time is spent in phase 1, creating the architecture. At this phase of the process, no board is touched, and no software is coded. It is about putting full attention, concentration and investigative skills into gathering information about the device to be developed, understanding what options exist, and documenting those findings.

---

3. What are the six stages in creating an architecture?

- *Stage 1*: Having a solid technical base;
- *Stage 2*: Understanding the architecture business cycle;
- *Stage 3*: Defining the architectural patterns and reference models;
- *State 4*: Creating the architectural structures;
- *Stage 5*: Documenting the architecture;
- *Stage 6*: Analysing and evaluating the architecture.

---

4. What are the ABCs of embedded systems? Draw and describe the cycle.

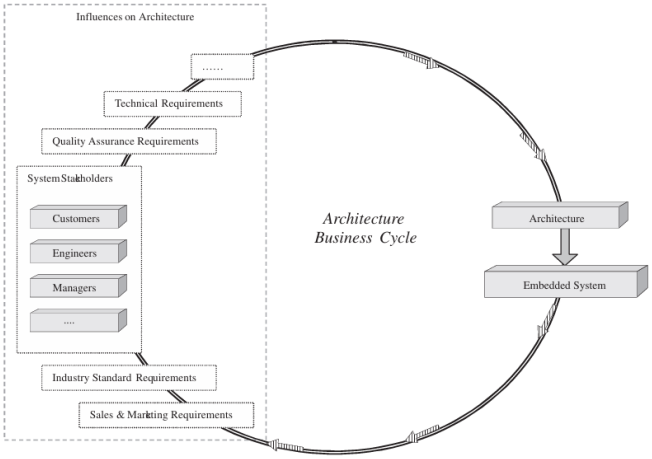

Figure 11-3: Architecture Business Cycle [11-2]

The ABCs of embedded systems are the many different types of influences that generate the requirements of the system, the requirements in turn generate the architecture, the architecture then produces the system, and the resulting system in turn provides requirements and capabilities back to the organization for future embedded designs.

---

5. List and define the four steps of Stage 2 of creating the architecture.

- *Step 1*: Understanding that ABC influences drive the requirements of an embedded system, and that these influences are not limited to technical ones.
- *Step 2*: Specifically identifying all the ABC influences on the design, whether technical, business, political and/or social.
- *Step 3*: Engaging the various influences as early as possible in the design and development lifecycle and gathering the requirements of the system.
- *Step 4*: Determining the possible hardware and/or software elements that would meet the gathered requirements.

---

6. Name four types of influences on the design process of an embedded system.

- *Business*: Sellability, Time-to-Time Market, Costs
- *Technical*: Performance, Reliability, Scalability, Portability, Availability
- *Industry*: Standards
- *Quality Assurance*: Testability, Availability, Schedule
- *Customers*: Cost, Performance

---

7. Which method is least recommended for gathering information from ABC influences? A. finite-state machine models / B. scenarios / C. by phone / D. in an e-mail / E. All of the above. Answer **D) in an email**

---

8. Name and describe four examples of general ABC features from five different influences.

Refer Q6

---

9. What is a prototype? How can a prototype be useful?

A useful tool in understanding, capturing, and modelling system requirements is through utilizing a *system prototype*, a physically running model containing some combination of system requirements. A prototype can be used to define hardware and software elements that could be implemented in a design, and indicate any risks involved in using these elements. Using a prototype in

---

---

conjunction with the general ABC features allows you to accurately determine early in the project what hardware and software solutions would be the most feasible for the device to be designed.

---

10. What is the difference between a scenario and a tactic?

In the context of Embedded Systems, a scenario often describes a potential usage or operational environment in which the embedded system will function. It could involve factors such as user interactions, environmental conditions, or system inputs. Conversely, a tactic in Embedded Systems refers to a specific implementation or approach used to accomplish a particular function or task within the embedded system's operation. While a scenario outlines the broader context, a tactic focuses on the detailed methods or strategies employed within that context to achieve desired system behaviour or functionality.

---

11. []

---

12. [T/F] A requirement can have multiple tactics. **True**.

Requirements specify what the system must do or accomplish, while tactics outline how those requirements will be met. Depending on the complexity of the requirement or the system itself, multiple tactics may be employed to fulfil it effectively. Each tactic represents a different approach or strategy for satisfying the requirement, offering flexibility in design and implementation.

---

13. What is the difference between an architectural pattern and a reference model?

An architectural pattern is a high-level blueprint or profile of an embedded system, outlining the arrangement and interaction of software and hardware components. It describes the types of elements present in the system, their functions, their interrelationships, and external interfaces.

On the other hand, a reference model represents a specific topological layout of the hardware and software elements within the embedded system. It serves as a concrete representation of the system's architecture, illustrating how different components are structured and connected.

---

14. What is the “4+1” model? Why is it useful? List and define structures that correspond to the “4+1” model.

The “4+1” model states that a system architect should create five concurrent structures per architecture at the very least, and each structure should represent a different viewpoint of the system. What is literally meant by “4+1” is that four of the structures are responsible for capturing the various requirements of the system. The fifth structure is used to validate the other four, ensuring that there are no contentions between the structures and that all structures describe the exact same embedded device, from their various viewpoints.

- *Structure 1:* A logical modular structure showcases key functional hardware and software elements, outlining their interrelationships to meet system requirements, aiding in system building and integration.
- *Structure 2:* A process structure reflects concurrency and synchronization, demonstrating how an OS meets nonfunctional requirements, such as performance and resource availability.
- *Structure 3:* A development structure maps hardware and software into the development environment, aiding buildability and considering constraints like IDEs and programming complexity.
- *Structure 4:* A deployment/physical structure illustrates how software maps onto hardware, ensuring nonfunctional requirements like hardware resource availability and reliability are met, defining hardware requirements based on software needs.

---

15. What is the process for documenting an architecture? How can a particular structure be documented?

- *Step 1:* Create a comprehensive table of contents outlining the architecture documentation, including an overview of the system, supported requirements, structure definitions, interrelationships, documentation layout, and modelling techniques.
- *Step 2:* Develop individual documents for each structure detailing supported requirements, design rationale, constraints, and graphical representations of structural elements, incorporating both graphical and nongraphical representations and outlining any external interfaces or protocols.
- *Step 3:* An architecture glossary. This document lists and defines all the technical terms used in all of the architectural documentation.

---

16. List and define two common approaches for analysing and evaluating an architecture? Give at least five real-world examples of either.

1. *Architecture Level Prediction of Software Maintenance (ALPSM):* This approach evaluates maintainability through scenarios, assessing how well the architecture supports future maintenance activities.
2. *The Architecture Tradeoff Analysis Method (ATAM):* A quantitative quality attribute approach that identifies problem areas and technical implications of an architecture through questions and measurements. It can be applied to evaluate various quality attributes.

Real-world examples:

1. *Maintainability of Operating Systems:* ALPSM can predict the ease of maintaining an operating system architecture over its lifecycle.
2. *Evaluation of Web Application Architecture:* ATAM can assess the scalability and performance implications of a web application architecture, helping to optimize its design.
3. *Cost-Benefit Analysis for Cloud Architectures:* CBAM, an extension of ATAM, can analyze the economic implications of adopting specific cloud architectures.
4. *Evaluation of IoT Device Architecture:* ISO/IEC 9126-1 thru 4 standards can be used to assess the reliability, usability, and efficiency of an IoT device architecture.
5. *Real-time Embedded System Design:* Rate Monotonic Analysis (RMA) evaluates the real-time behaviour of an embedded system architecture, ensuring timely task execution and system reliability.

---

17. What is the difference between a qualitative and quantitative quality attribute approach?

In embedded systems, a qualitative quality attribute approach involves evaluating system characteristics based on subjective judgments or descriptions, providing an understanding of attributes like reliability or power consumption without assigning numerical values.

Conversely, a quantitative quality attribute approach entails measuring attributes using numerical metrics, facilitating precise analysis and comparison of aspects such as response time or memory usage, thereby enabling objective assessment and optimization of embedded systems.

---

18. What are the five steps introduced in the text as a method by which to review an architecture?

- *Step 1:* The evaluation team receives architecture documentation, briefed on the evaluation process and content.
- *Step 2:* Architectural approaches and patterns are listed based on team feedback after analyzing the documentation.
- *Step 3:* The team and architects agree on scenarios derived from system requirements, prioritizing based on importance and implementation difficulty.
- *Step 4:* Emphasis is placed on evaluating high-risk and crucial scenarios.
- *Step 5:* Evaluation results encompass agreed requirements, benefits, risks, strengths, problems, and recommended design changes.

CSPC61, EMBEDDED SYSTEMS AND ARCHITECTURE
CHAPTER-12: The Final Phases of Embedded Design: Implementation and Testing

1. What is the difference between a host and a target?
   A development environment is typically made up of a target (the embedded system being designed) and a host (a PC, Sparc Station, or some other computer system where the code is actually developed).

2. What high-level categories do development tools typically fall under?
   The key development tools are:
   Utility tool : Example- editors (for writing source code), VCS (Version Control Software) that manages software files, ROM burners that allow software to be put onto ROMs
   Translation tool : Convert code (that a developer intends for the target) into a form the target can execute
   Debugging tools : Can be used to track down and correct bugs in the system

3. [T/F] An IDE is used on the target to interface with the host system.
   False. Source code is typically written with a tool such as a standard ASCII text editor, or an Integrated Development Environment (IDE) located on the host platform.

4. What is CAD?
   It is a software commonly used by hardware engineers to simulate circuits at the electrical level in order to study a circuit's behavior under various conditions before they actually build the circuit.

5. In addition to CAD, what other techniques are used to design complex circuits?
   SPICE(Simulation Program with Integrated Circuit Emphasis), PSpice

6. A) What is a preprocessor?
   Preprocessing is an optional step that occurs either before the translation or interpretation of source code. Its functionality is commonly implemented by a preprocessor. The preprocessor's role is to organize and restructure the source code to make translation or interpretation of this code easier.

   B) Provide a real-world example of how a preprocessor is used in relation to a programming language.

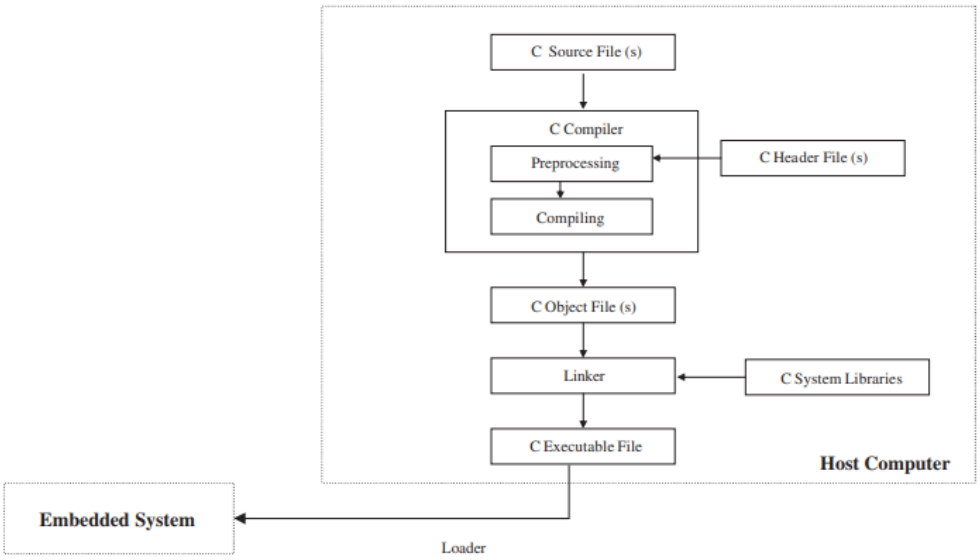

Figure 12-5: C example compilation/linking steps and object file results

7. [T/F] A compiler can reside on a host or a target, depending on the language.
   True. Most compilers are located on the programmer's host machine. Some languages, such as Java and scripting languages, have compilers or interpreters located on the target.

8. What are some features that differentiate compiling needs in embedded systems versus in other types of computer systems?
   (1) embedded designs can contain several different types of physical memory; (2) they typically have a limited amount of memory compared to other types of computer systems;
   (3) memory can typically become very fragmented and defragmentation functionality is not available out-of-the-box or too expensive; and
   (4) certain types of embedded software may need to be executed from a particular memory location.

9. A) What is an object file?
   After all the compilation on the programmer's host machine is completed, the remaining target code file is commonly referred to as an object file. It can contain anything from machine code to Java byte code, depending on the programming language used.

   B) What is the difference between a loader and a linker?
   Linker integrates this object file with any other required system libraries, creating what is commonly referred to as an executable binary file. This executable is either directly transferred onto the board's memory or ready to be transferred to the target embedded system's memory by a loader.

CSPC61, EMBEDDED SYSTEMS AND ARCHITECTURE
CHAPTER-12: The Final Phases of Embedded Design: Implementation and Testing

10. A) What is an interpreter?
    An interpreter translates source code into object code one instruction at a time. The resulting object code is then executed immediately. The process is called interpretation.

    B) Name three real-world languages that require an interpreter.
    Python, Javascript, PHP

11. An interpreter resides on: B) both host and target

12. A) What is debugging?
    Task of locating and fixing errors within the system

    B) What are the main types of debugging tools?
    Hardware, Software and Manual

    C) List and describe four real-world examples of each type of debugging tool.
    Refer table pg 552 in book.

13. What are five of the cheapest techniques to use in debugging?
    Manual Tools are Readily available, free or cheaper than other solutions, effective, simpler to use.
    5 of them are: Print Statements, Dump files, Counters/Timers, Fast Display, Output ports

14. Boot code is: C) Software that starts-up the board.
    When power is applied to an embedded board (because of a reset), start-up code, also referred to as boot code, bootloader, bootstrap code, or BIOS (basic input output system) depending on the architecture, in the system's ROM is loaded and executed by the master processor.

15. What is the difference between debugging and testing?
    Goals of debugging are to actually fix discovered bugs. Debugging typically occurs when the developer encounters a problem in trying to complete a portion of the design, and then typically tests to-pass the bug fix.
    With testing, on the other hand, bugs are discovered as a result of trying to break the system, including both testing-to-pass and testing to-fail, where weaknesses in the system are probed.
16. A) List and define the four models under which testing techniques fall.
    static black box testing
    static white box testing
    dynamic black box testing
    dynamic white box testing

Black box testing occurs with a tester that has no visibility into the internal workings of the system (no schematics, no source code, etc.). Black box testing is based on general product requirements documentation, as opposed to white box testing (also referred to clear box or glass box testing) in which the tester has access to source code, schematics, and so on. Static testing is done while the system is not running, whereas dynamic testing is done when the system is running.

B) Within each of these models, what are five types of testing that can occur?
unit/module testing (incremental testing of individual elements within the system)
compatibility testing (testing that the element doesn't cause problems with other elements in the system)
integration testing (incremental testing of integrated elements), system testing (testing the entire embedded system with all elements integrated)
regression testing (rerunning previously passed tests after system modification),
manufacturing testing (testing to ensure that manufacturing of system didn't introduce bugs)

17. [T/F] Testing-to-pass is testing to insure that system minimally works under normal circumstances.
    True

18. What is the difference between testing-to-pass and testing-to-fail?
    tests-to-pass tests only to ensure the system minimally works under normal circumstances whereas testing-to-fail tries to break the system and weaknesses in the system are probed.

19. Name and describe four general areas of law under which a customer can sue for product problems.
    Breach of Contract (i.e., if bug fixes stated in contract are not forthcoming in timely manner)
    Breach of Warranty and Implied Warranty (i.e., delivering system without promised features)
    Strict and Negligence liability for personal injury or damage to property (i.e., bug causes injury or death to user)
    Malpractice (i.e., customer purchases defective product)

20. [T/F] Once the embedded system enters the manufacturing process, the design and development team's job is done.
    False
    The responsibilities of the engineering team last throughout the lifecycle of the device, and do not end when the embedded system has been deployed to the field.