Embedded Systems

Introduction And Theoretical Talk

About the Speaker

- BE, SBME'11
- Msc, CSE'17
- Embedded SW Engineer
  - Axxcelera Egypt Broadband Wireless
    - Physical Layer Implementation
    - Platform-dependent SW Engineer
    - Platform and Framework team leader
- Contact
  - ahmedsayed.elaraby@gmail.com

Outline

- In the beginning …
- Generic Embedded System
- Embedded System Development
- How To Become ESW Developer ??

In The Beginning …

Embedded Software Engineer !

- First, lets find out the difference between embedded system and the normal computer.
  - Well, normal computer is general purpose !
  - On the other-hand, an embedded system is embedded !!

- So the question is; how is “Embedded System” embedded ?

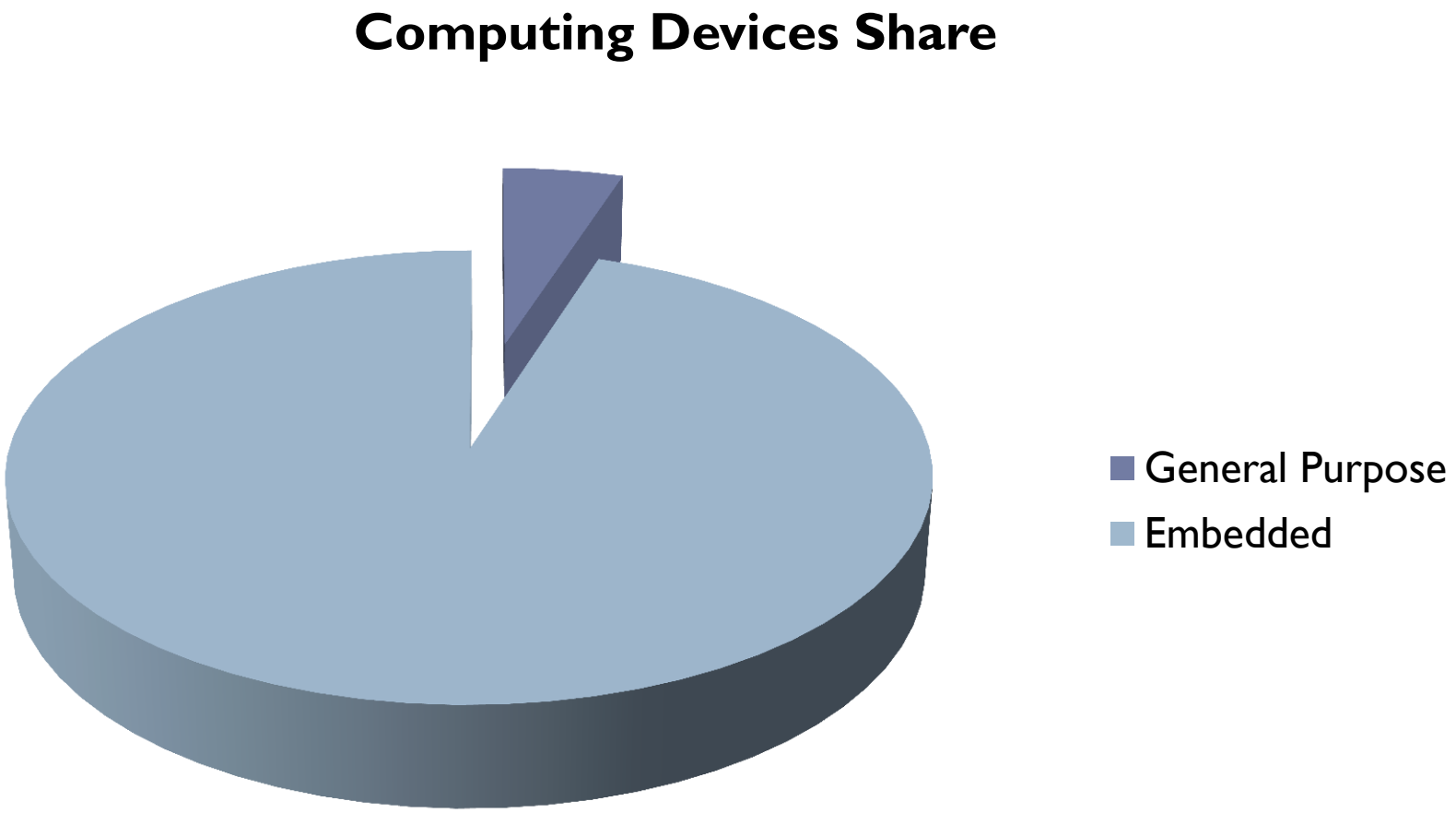
Embedded Software Engineer !

- Embedded means …
  - Purpose perspective: fits only for its designated purpose(s).
  - Embedded system is self-contained:
    - MP3 player vs. WMP

Embedded Software Engineer !

- Stand-alone, or part of other bigger system:
  - Information processing/control for other general of embedded systems.
  - Hard-drive controller:

PC vs. Not-PC



What do ESs do ?

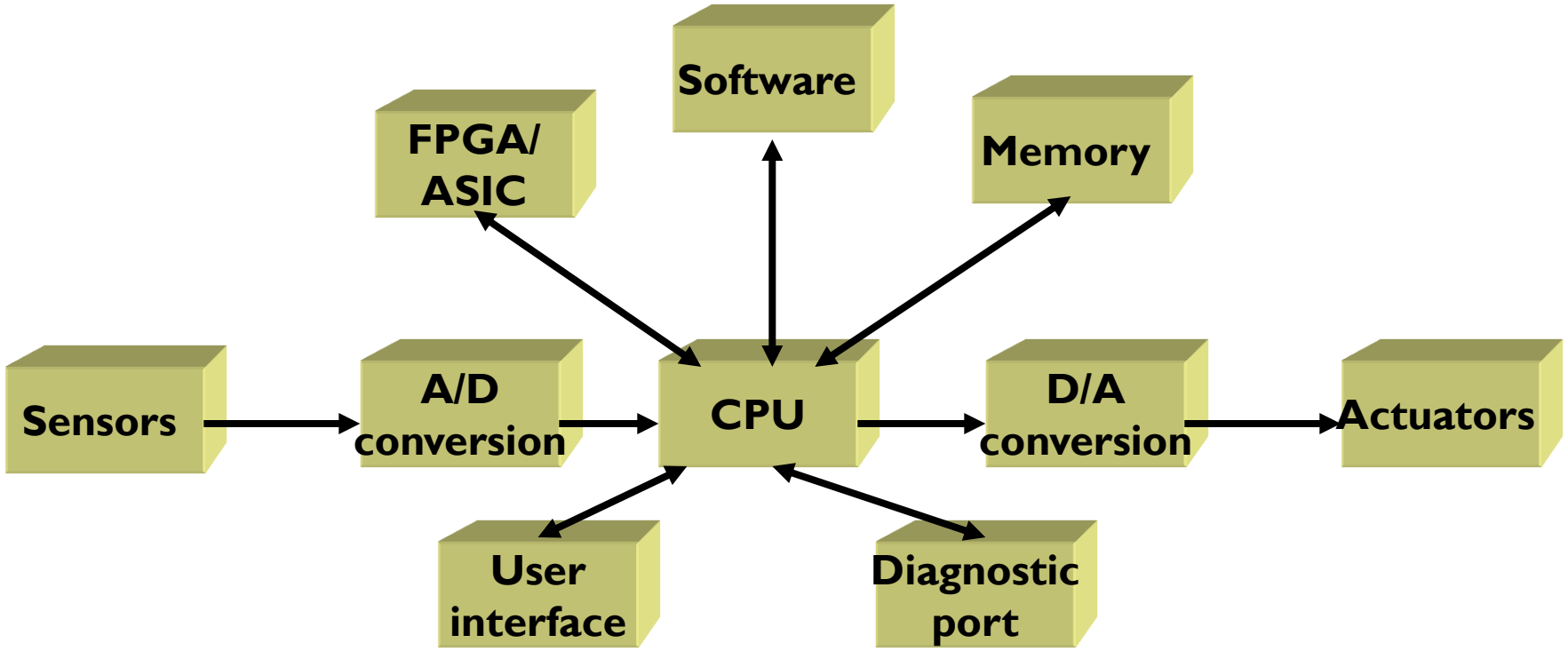
- Applications of an Embedded System (ES) can be classified under three categories:
  - Monitoring and Logging
    - Reading data from the environment via sensors, store and/or send these data.
  - Control
    - Control other systems by applying predefined control actions, FSM transitions or fuzzy control.
  - Processing
    - Transforming input data to more user-meaningful output via applying specific processing algorithm.

To Sum It Up …

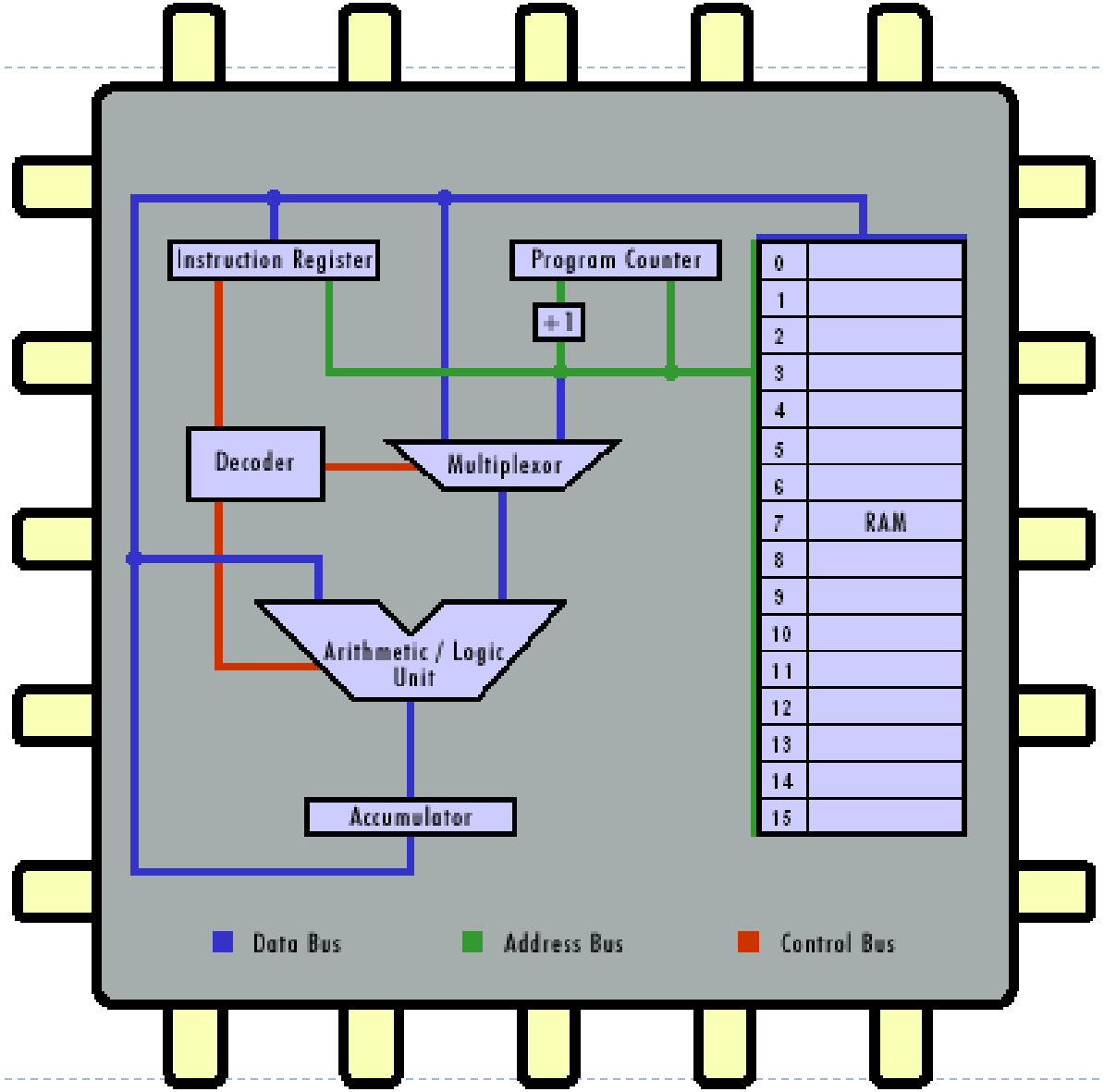
- Embedded system is a specific system that's optimized for a specific propose.
- Application executed via an ES is known before developing the system.
- Rare to entirely change the software (firmware) running on that system.
- Hardware requirements are precisely calculated, resources are limited and shared.
- Must ensure safety and reliability operations.

Generic Embedded System

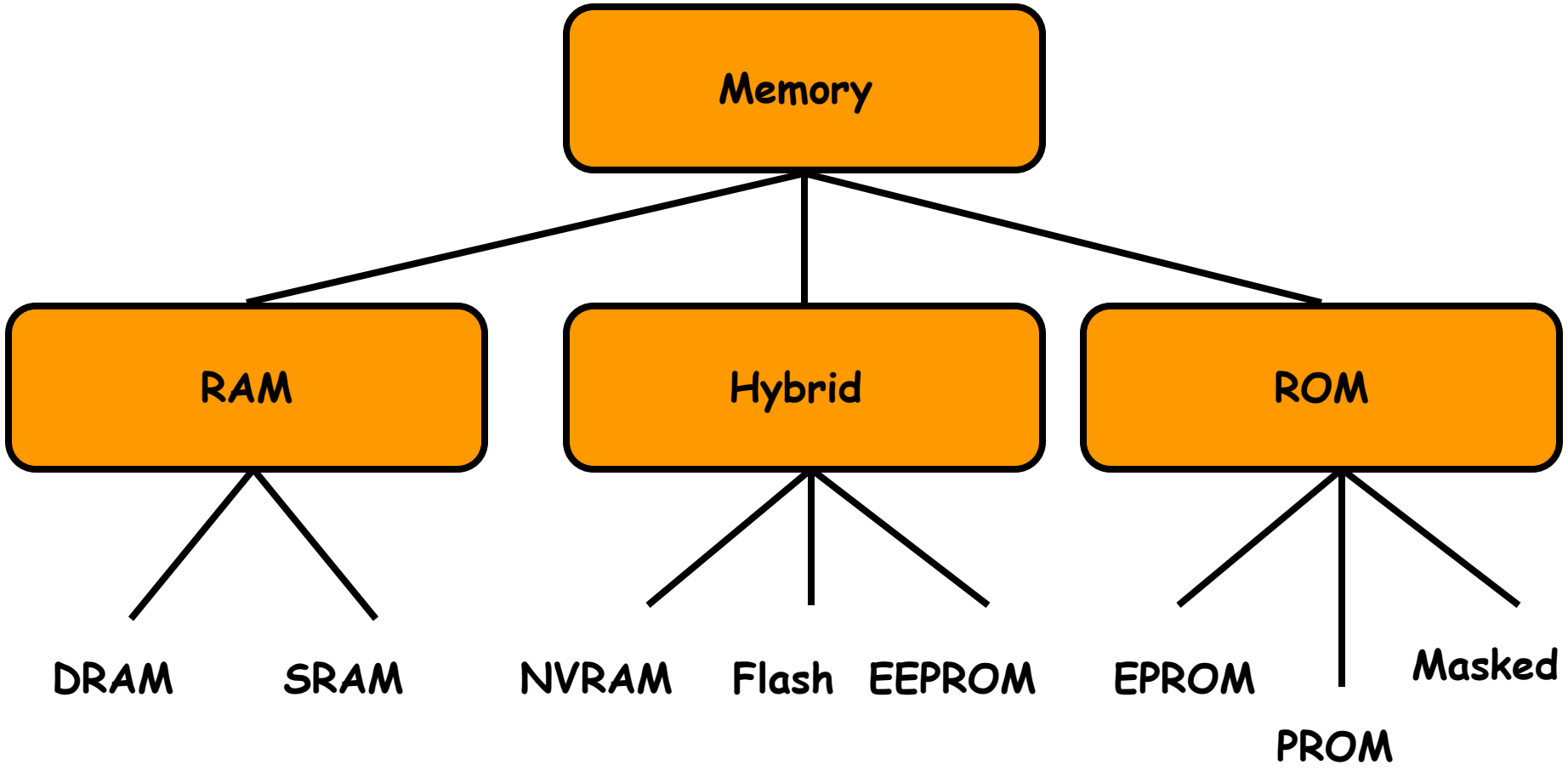
Basic Components



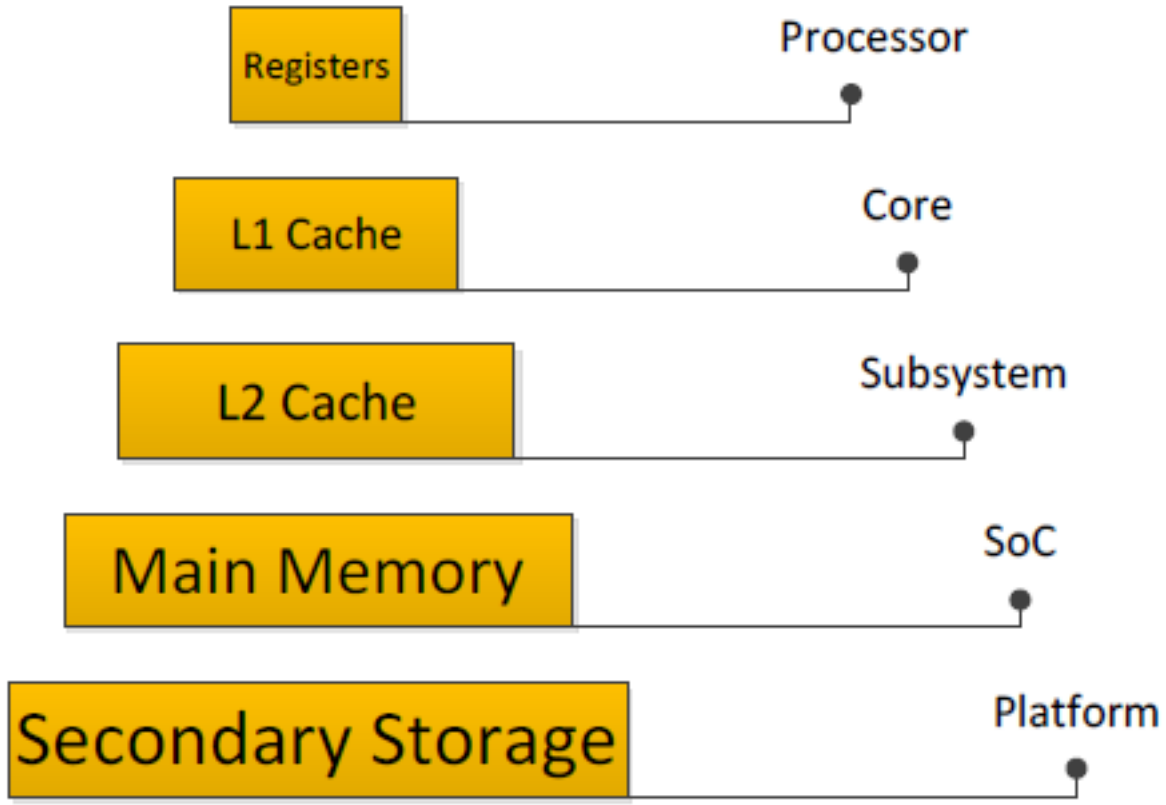
The CPU


Instruction Register
Program Counter
+1
Decoder
Multiplexor
Arithmetic / Logic Unit
Accumulator
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
RAM
Data Bus
Address Bus
Control Bus

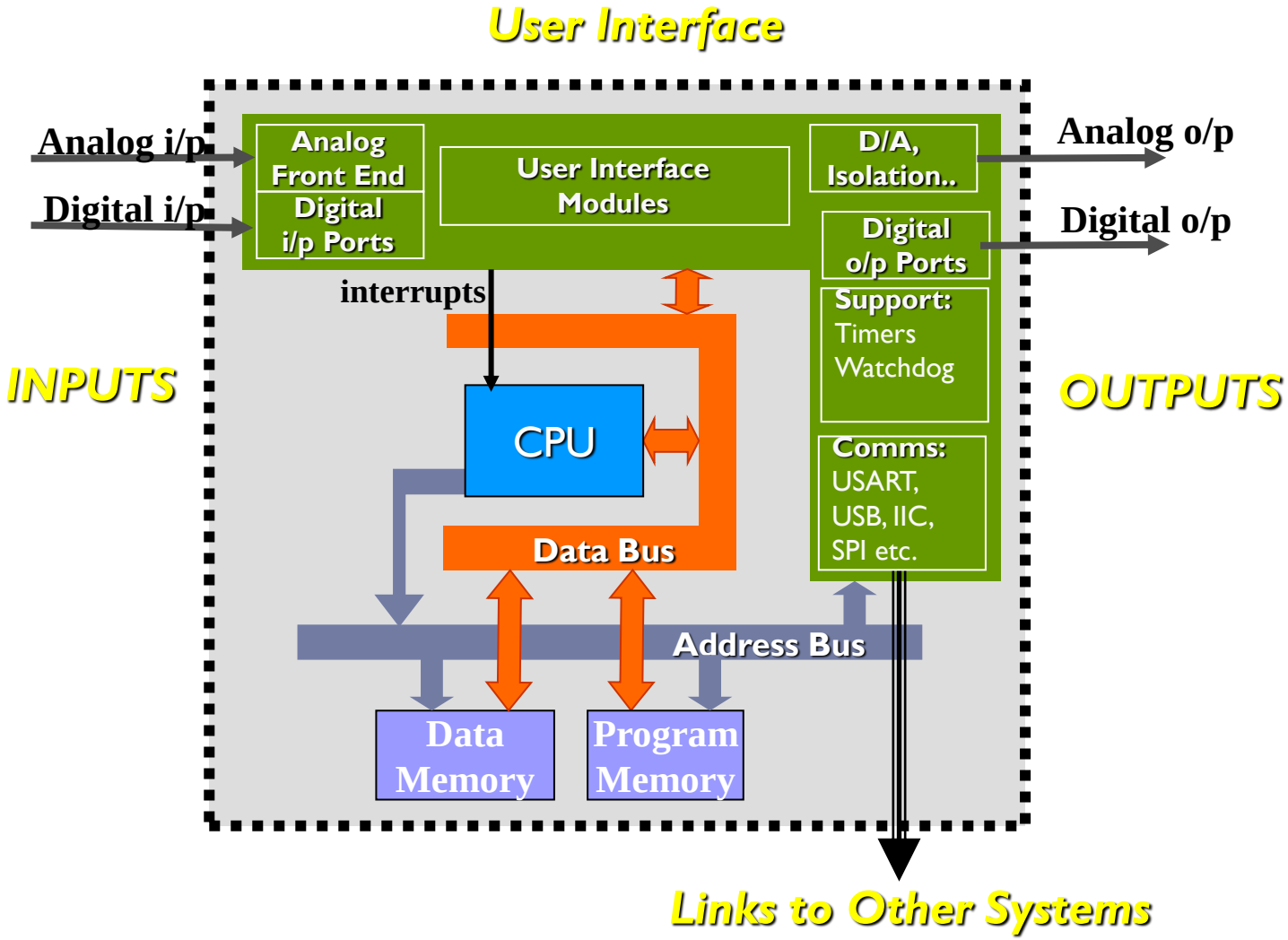
Memory


Memory
RAM
Hybrid
ROM
DRAM
SRAM
NVRAM
Flash
EEPROM
EPROM
PROM
Masked

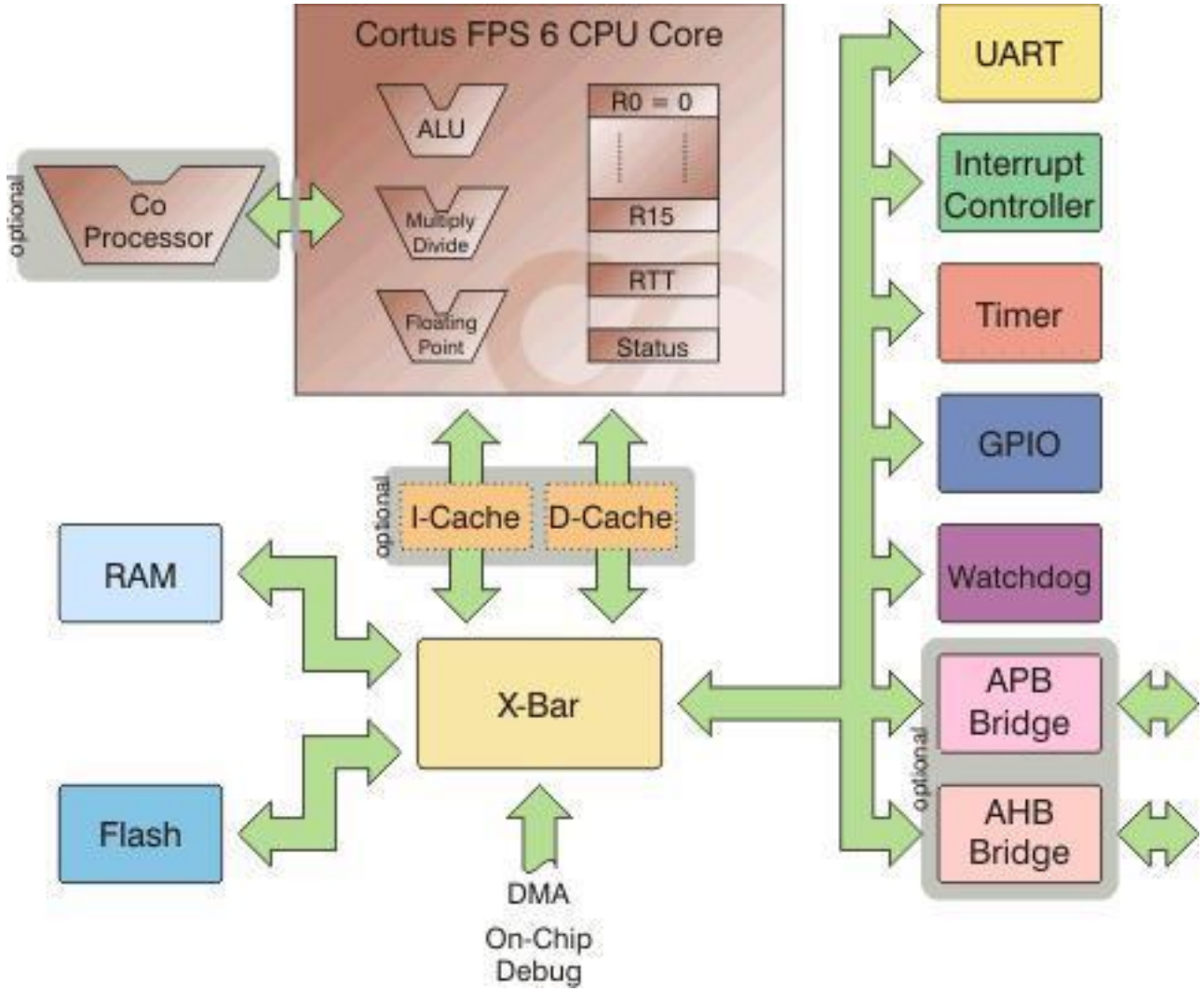
Memory


Registers
Processor
L1 Cache
Core
L2 Cache
Subsystem
Main Memory
SoC
Secondary Storage
Platform

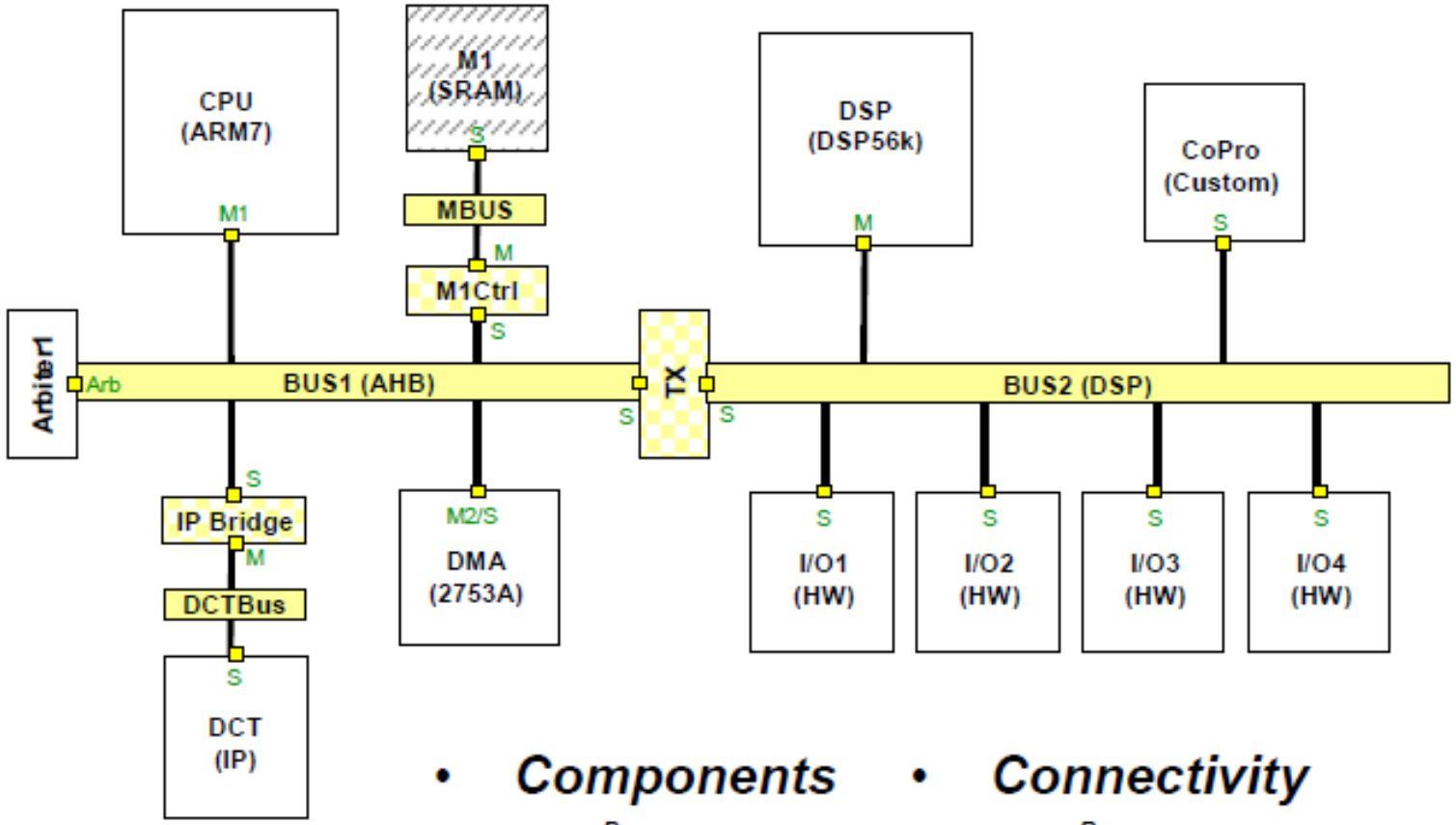
Typical Microcontroller



System


Cortus FPS 6 CPU Core
ALU
Multiply Divide
Floating Point
R0 = 0
R15
RTT
Status
optional
Co Processor
optional
I-Cache
D-Cache
RAM
Flash
X-Bar
DMA
On-Chip Debug
UART
Interrupt Controller
Timer
GPIO
Watchdog
APB Bridge
AHB Bridge
optional

System

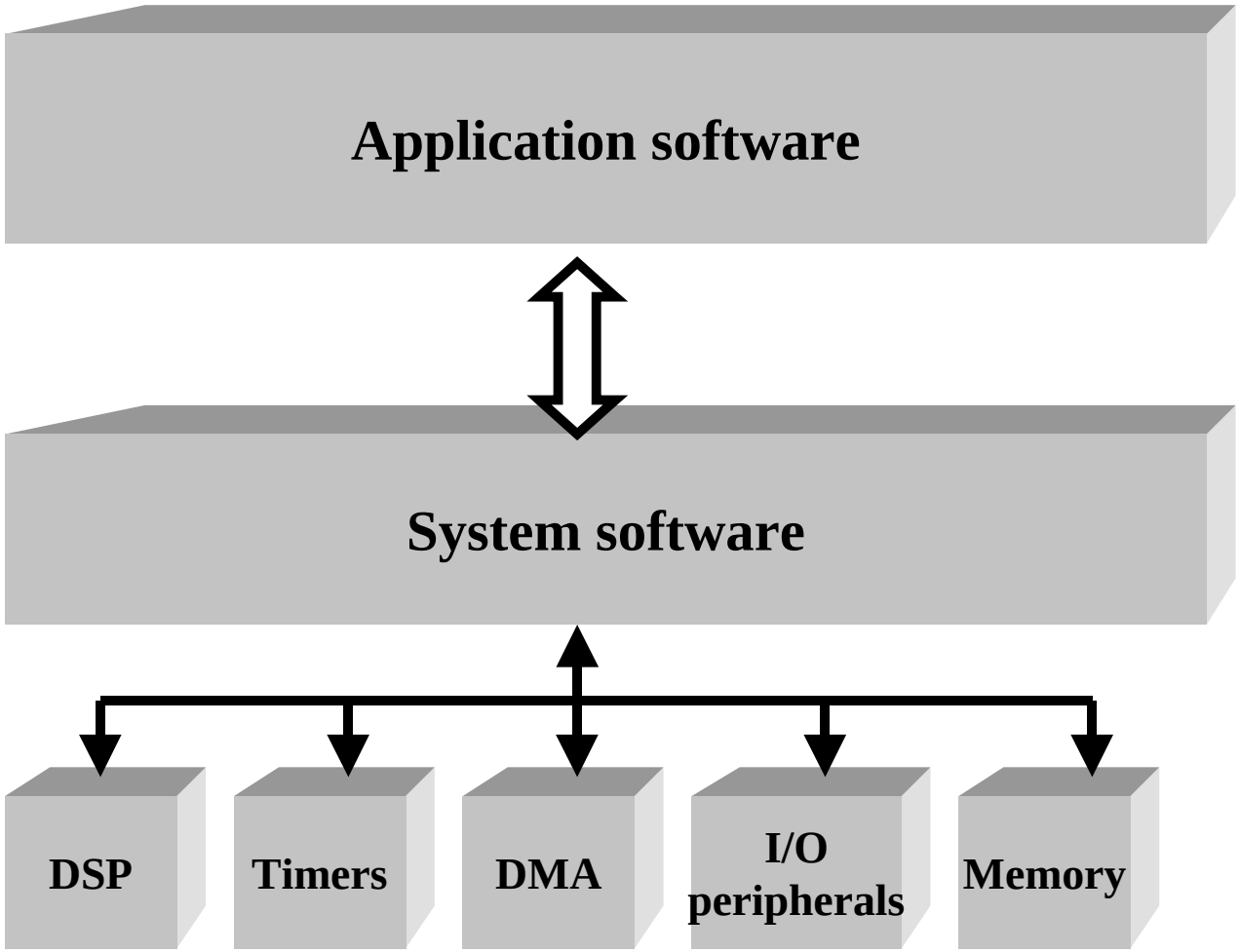


- **Components**
  - *Processors*
  - *Memories*
  - *IPs*
  - *Custom HW*
- **Connectivity**
  - *Buses*
  - *Bridges*
  - *Ports*

Software ?

- Programming language:
  - Mostly ‘C’
  - Why not JAVA/C# ?
    - C is structured language but not object oriented, this gives the C more control over low-level hardware and minimal memory (code-size and RAM usage) requirements.
  - So why not assembly ?
    - Assembly is not platform-independent, and very hard to program especially when the complexity of the program increases.

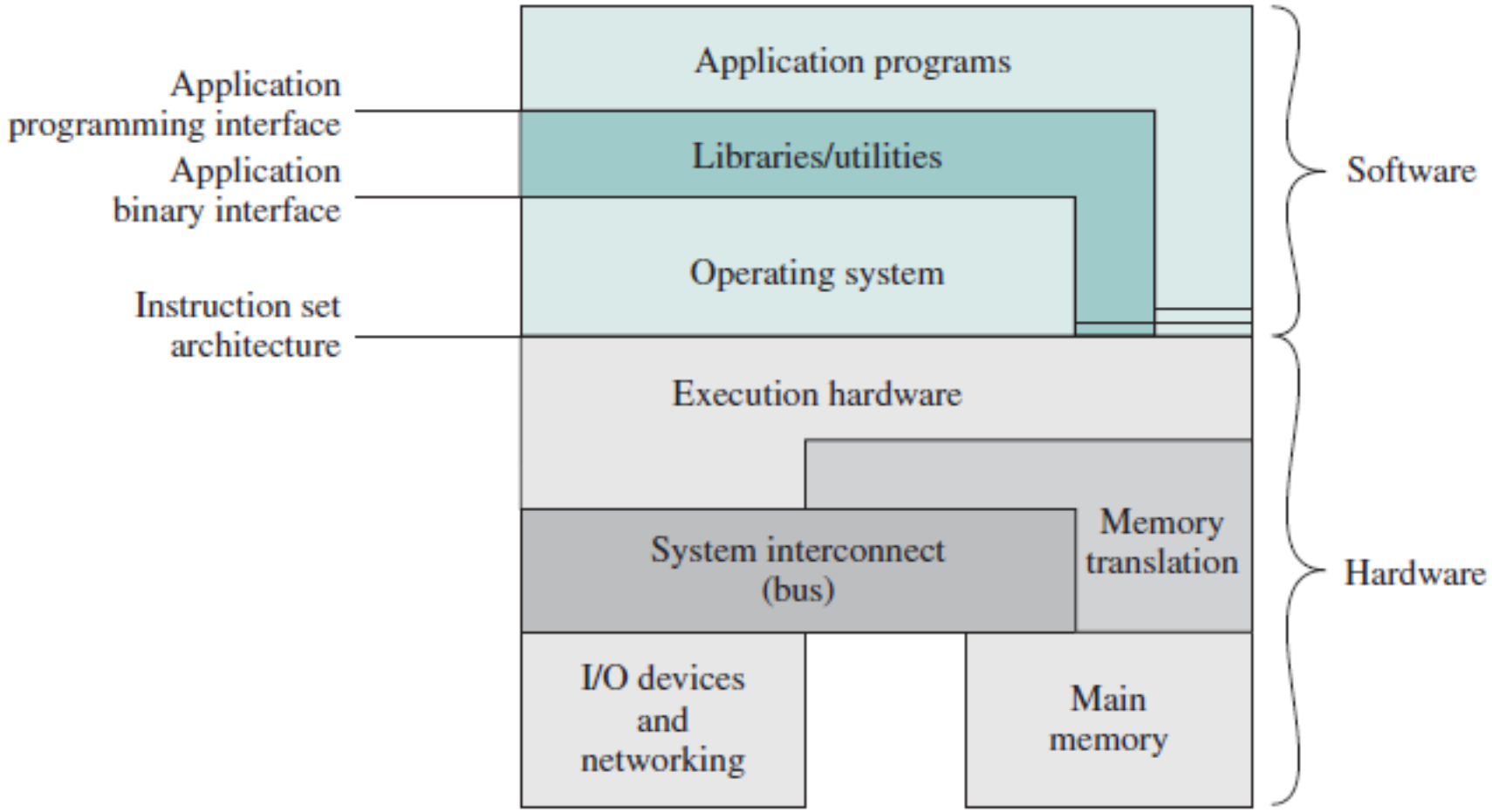
System software



System software

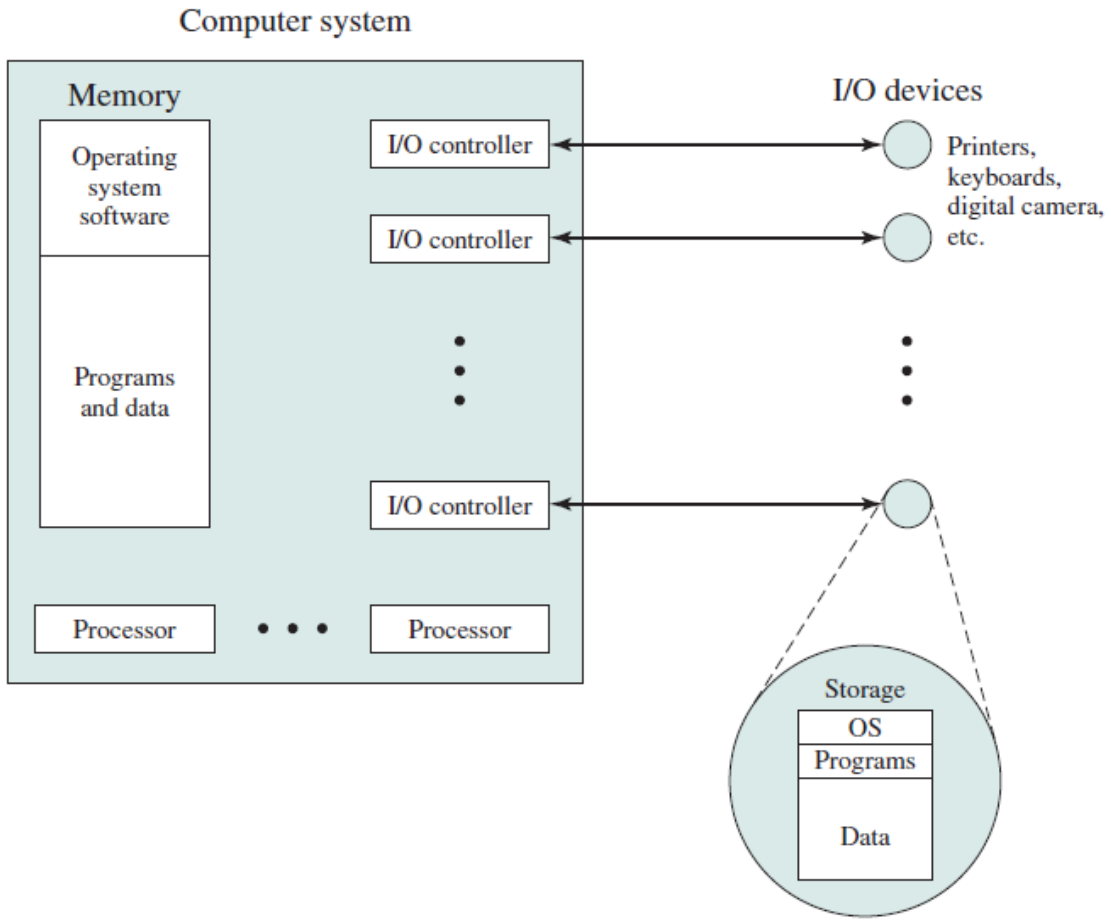
- Programmer can write his/her application that interacts directly with the processor and other hardware resources in the platform, or access these resources via other supporting software.

Operating system



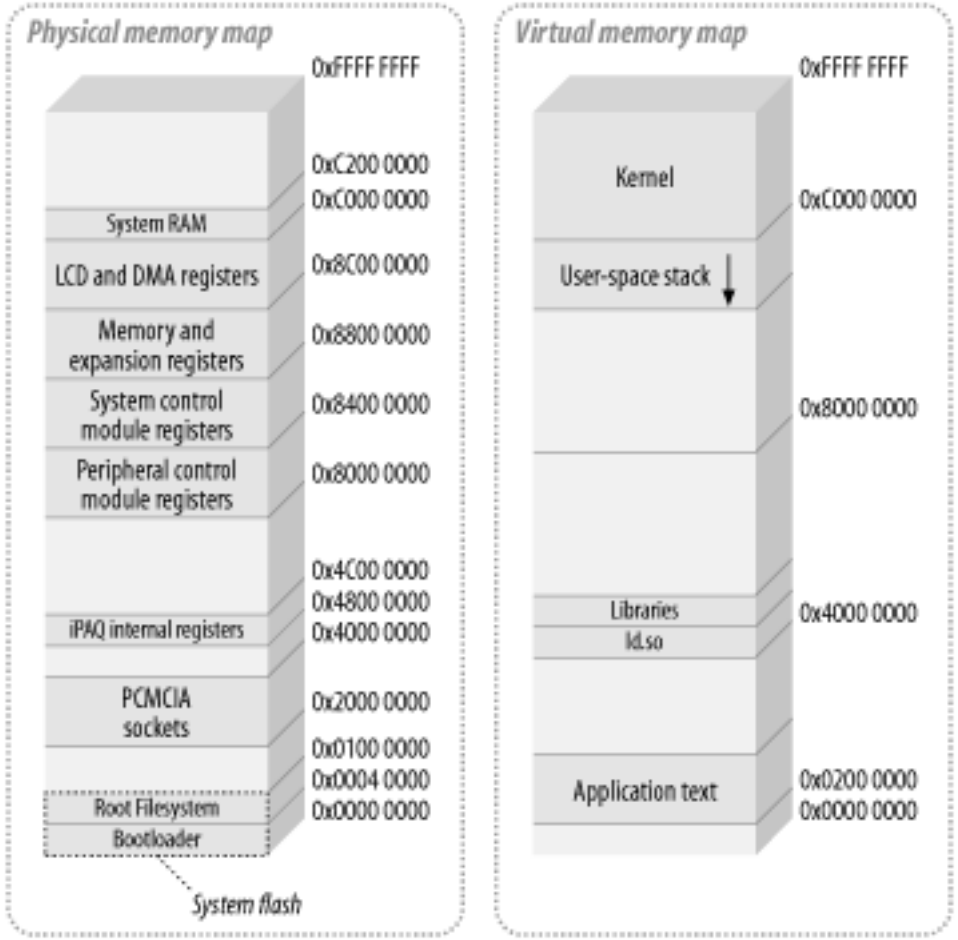
Operating system

- Resource Manager:



Board Support Package

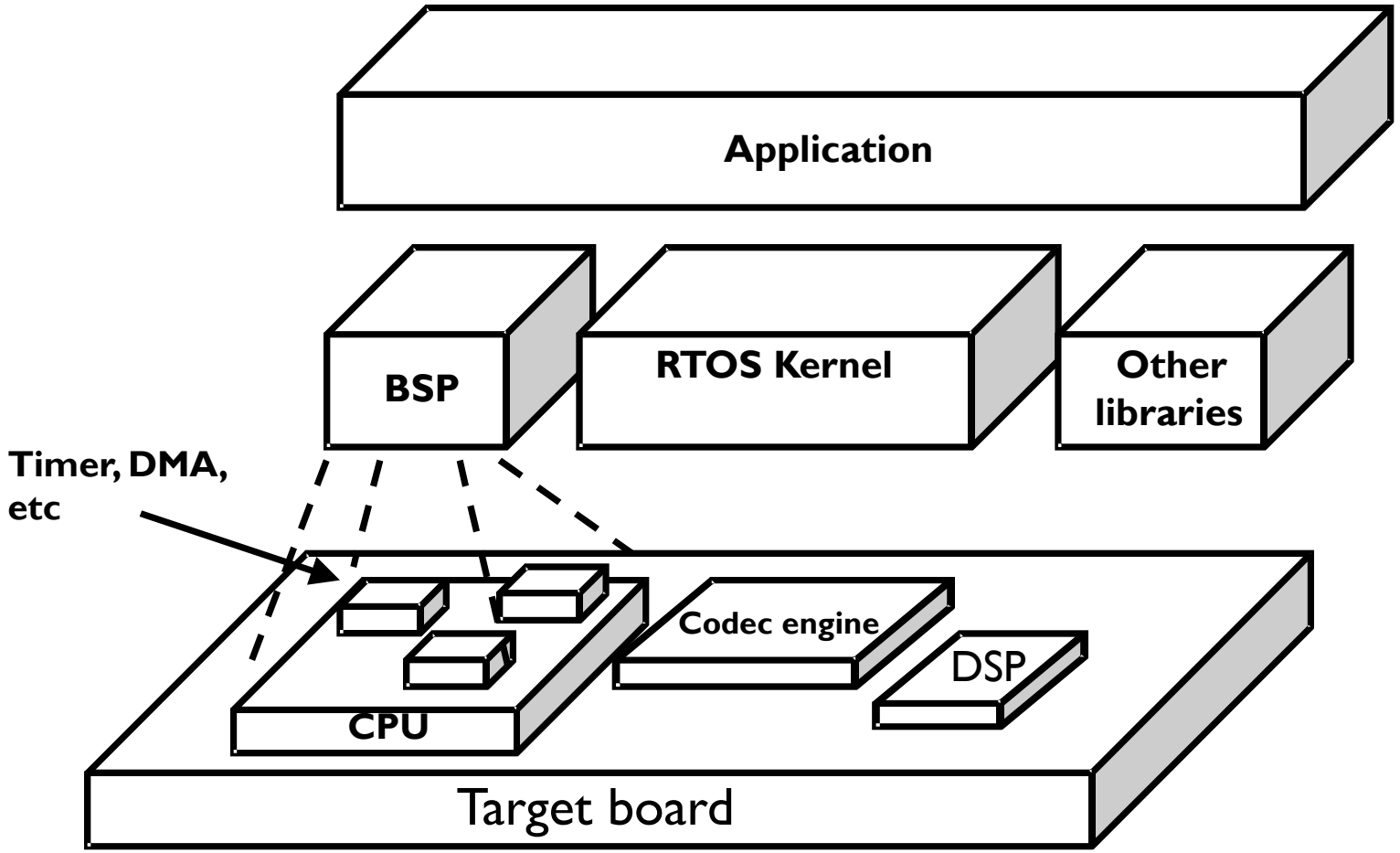
- Hardware memory-map:



Board Support Package

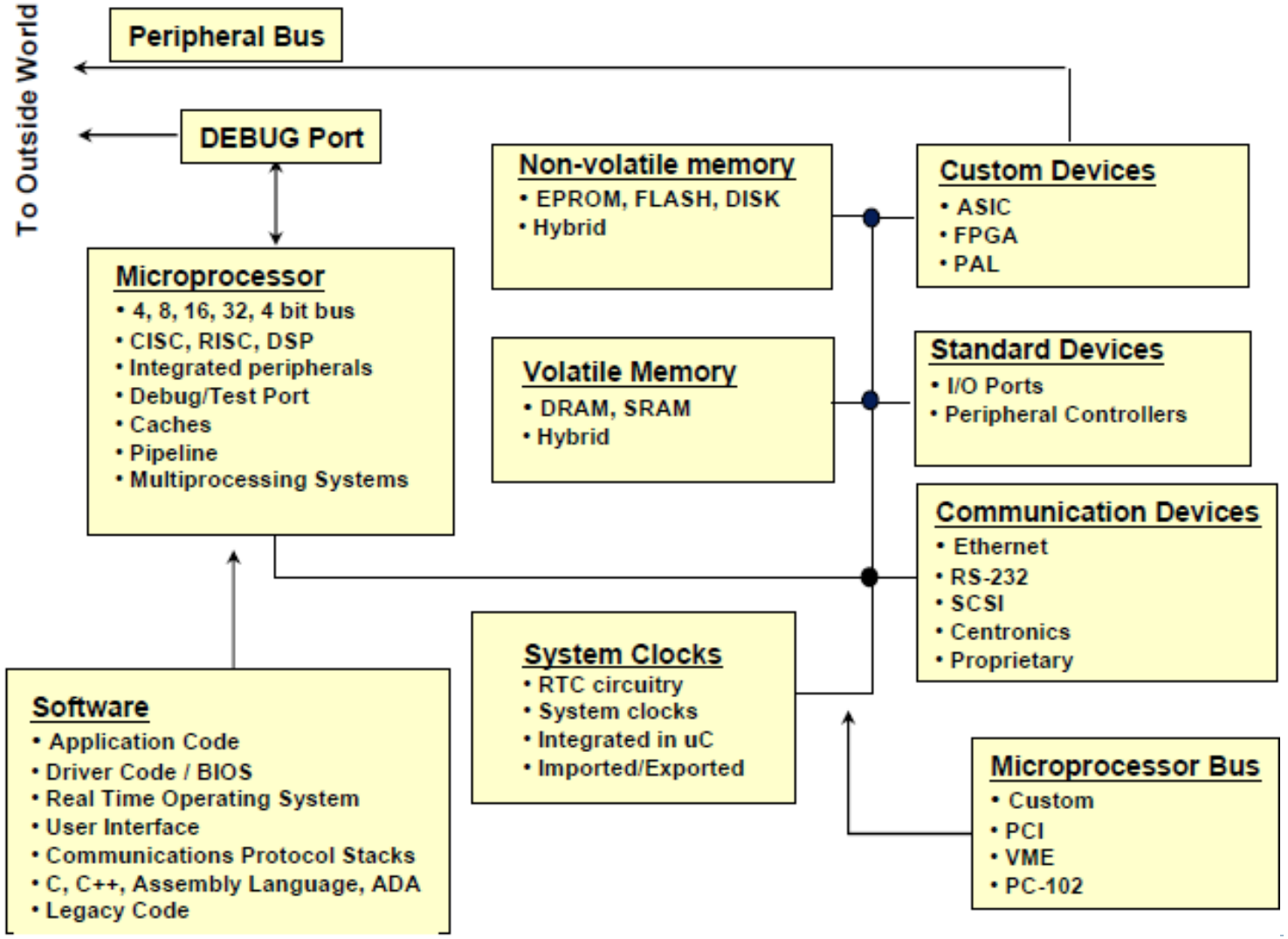
- Boot-loading.
- Platform initialization:
  - Processor
  - Bus
  - Memory Controller
  - Clock
  - ……
- HAL

Putting All Together

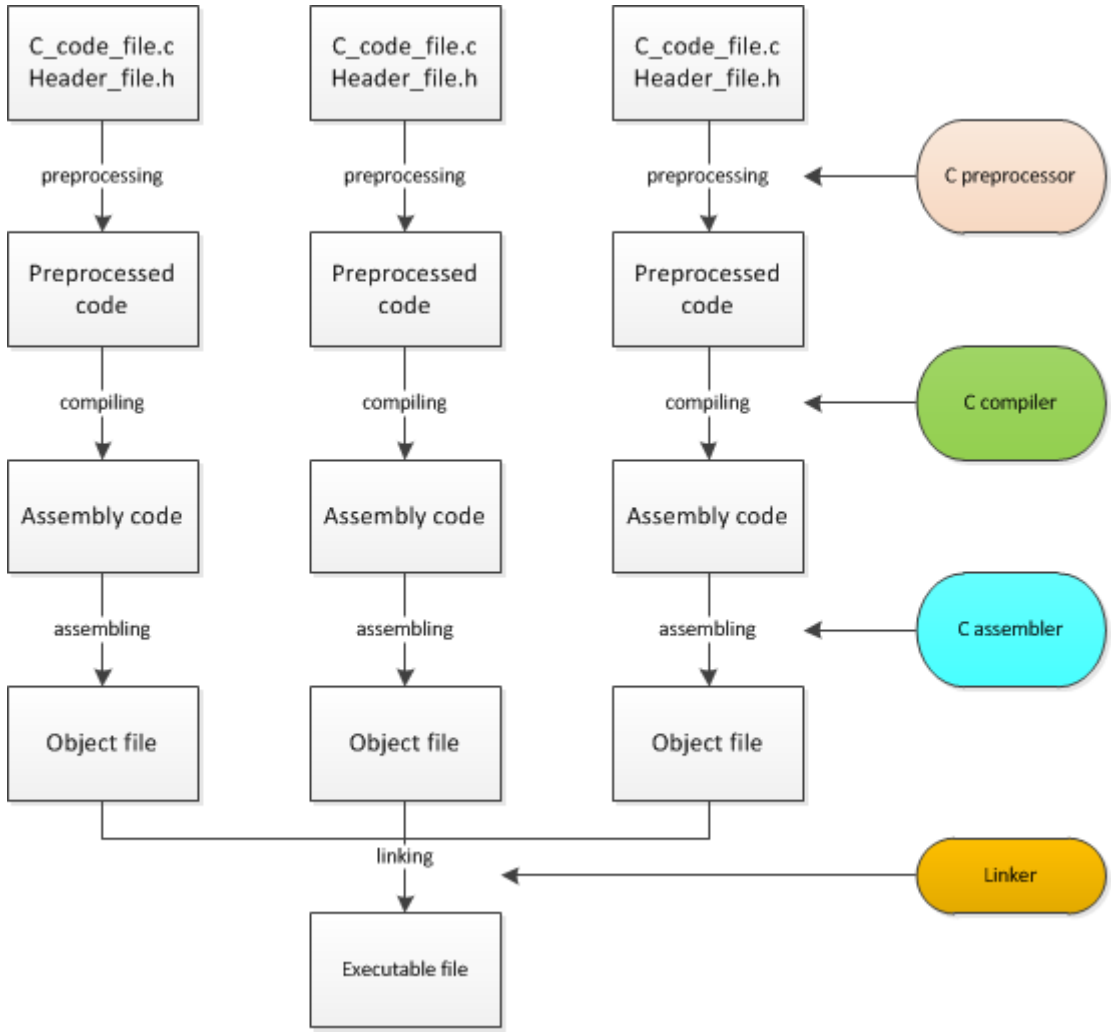


Embedded System Development

Typical Architecture



Building Steps …

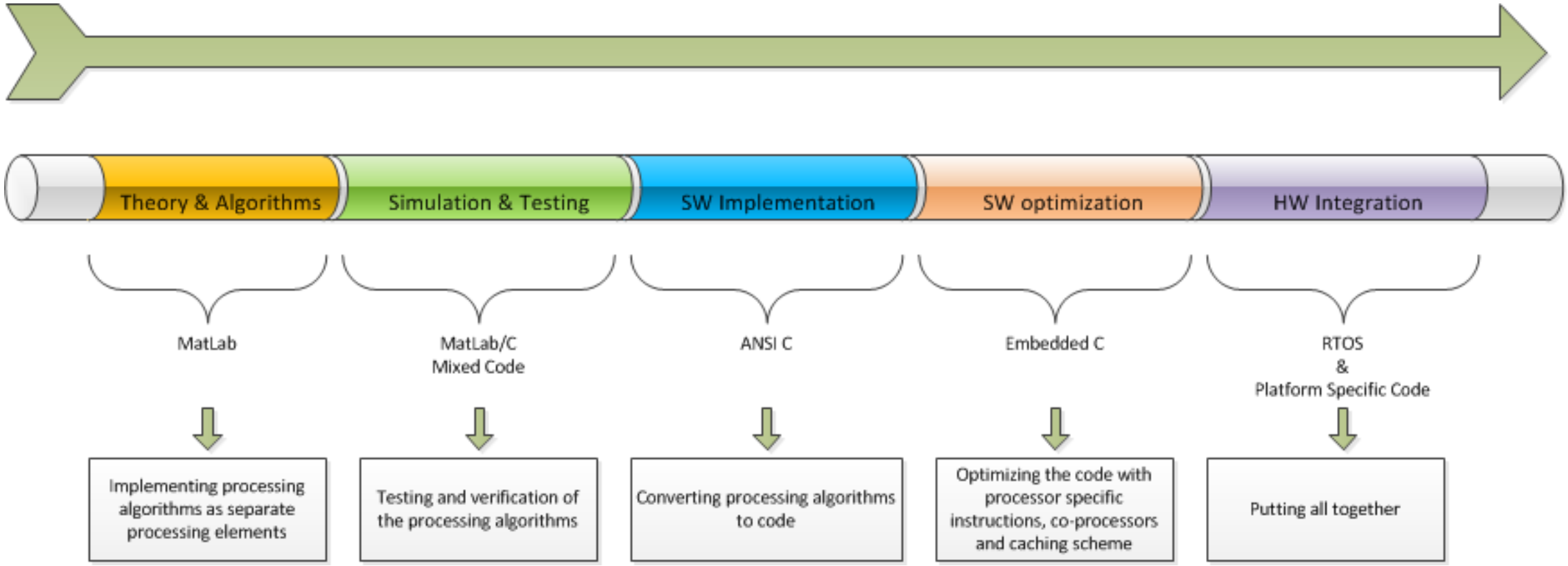


Debugging !!!

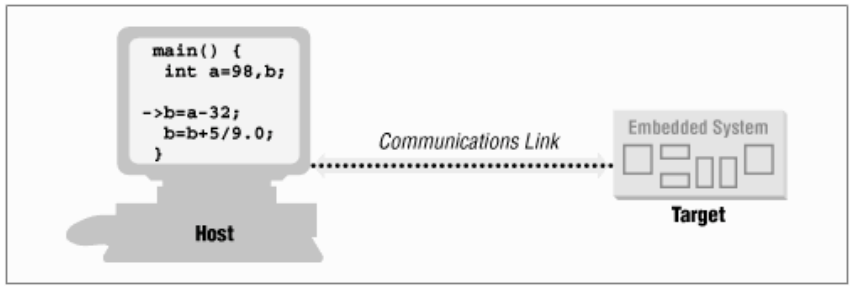
- Running the code, either in a simulated environment or on the target HW
  - Simulation provides assessment for the processing requirements
  - Usually, only the core is simulated, not the peripherals
- Using an IDE (Integrated Development Environment)
  - Handles the building process
  - Provides debugging facilities
    - Breakpoints
    - Watchpoints
    - Memory/register viewer
    - Profiling and tracing
    - …

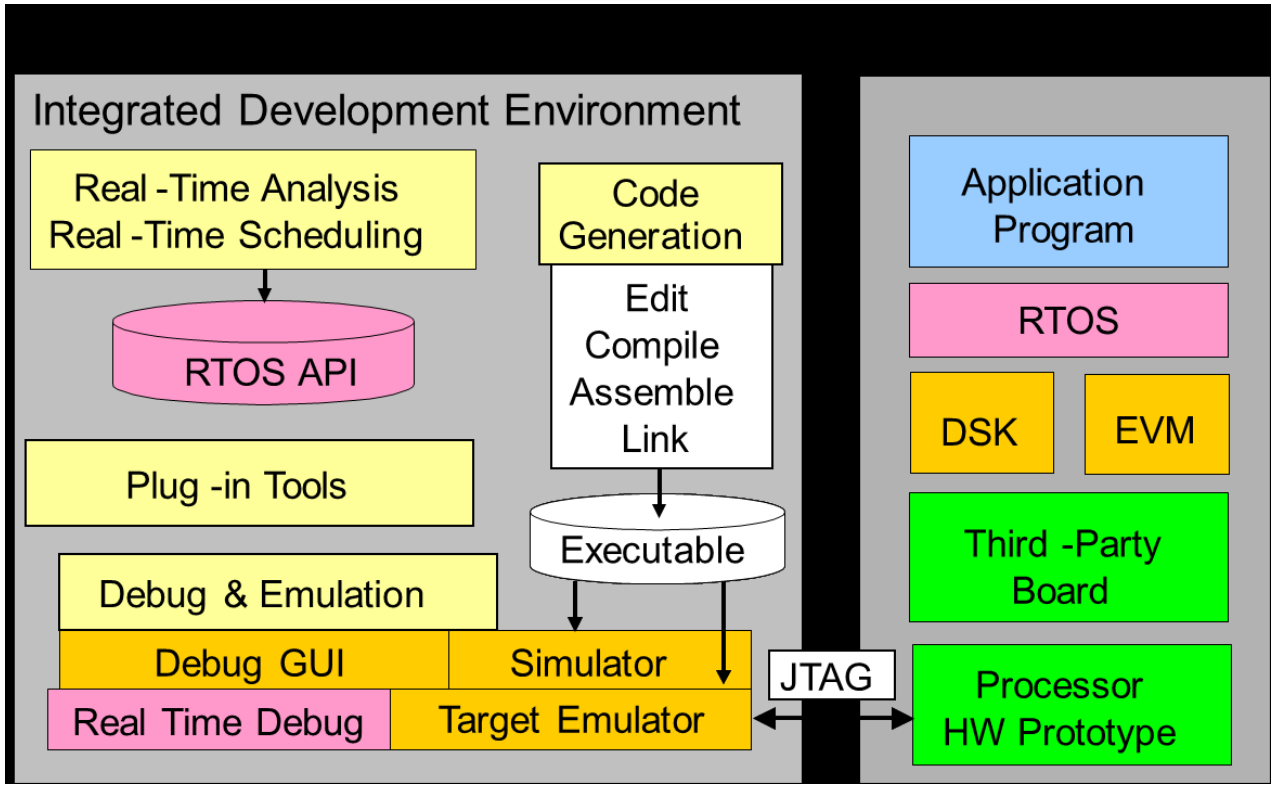
How To Become ESW Developer ??

Typical System Development Flow



Development Environment





Development Environment

How To Become ESW Developer ??

- Tough question !!!!
  - No easy, straight-forward answer :s
- But, I may have some advices to give …
  - Learn C .. Extensive C programming (NOT C++, but learn it too !)
  - Try to explore Linux
    - Basic shell commands
    - Write C programs, compile using GCC and Makefile, cross-compile
  - Write code .. Lots of code
    - Always think about memory and processing complexity
    - Try to use libraries
    - Write defensive code
    - Always, always comment on your code, and use VCS like GIT or Mercurial

How To Become ESW Developer ??

- Try to take online video lectures
- Start it out !
  - Try to get and work on embedded system, beginners may start with "Arduino" boards. Search for boards locally or via external supplier
  - Get in a community, like Arduino or Raspberry Pi
- Work in groups
  - Knowledge sharing is the key .. Divide and conquer strategy
- Search for idea to work on
  - Theoretical reading is boring !
  - Search for project ideas or your GP, even if you're 1st grader !

Resources

- Embedded Systems
  - Embedded Systems – Shape the World
  - The Hardware/Software Interface
  - Computer Architecture
- Linux
  - Introduction to Linux
  - Linux for Embedded Systems
- Arduino
  - The Arduino Platform and C Programming
- Python
  - Programming for Everybody