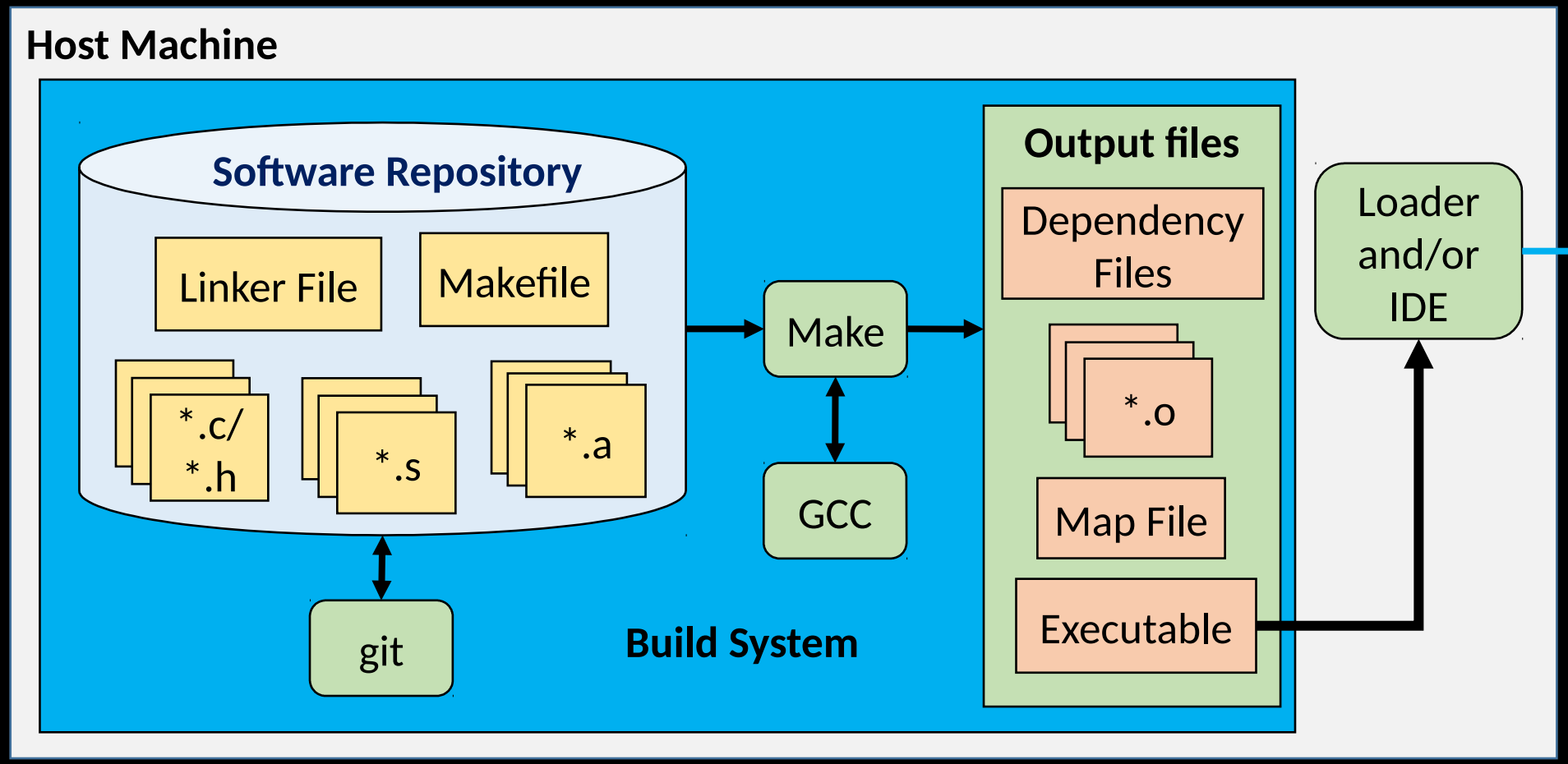
Architecture-Software Interface

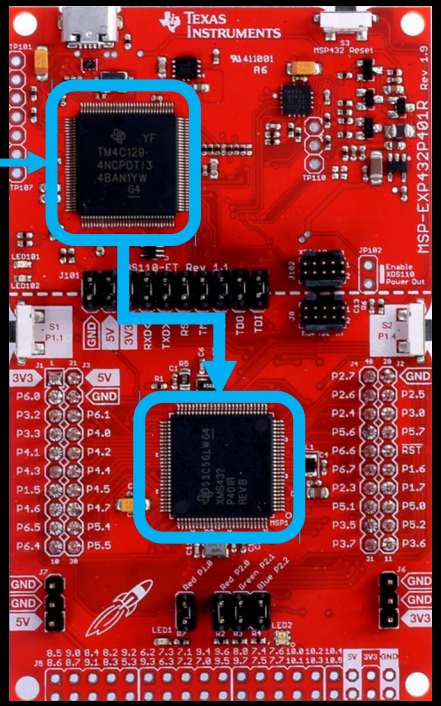
Embedded Software Essentials

C2M1V1

Embedded System Development Environment [S1]



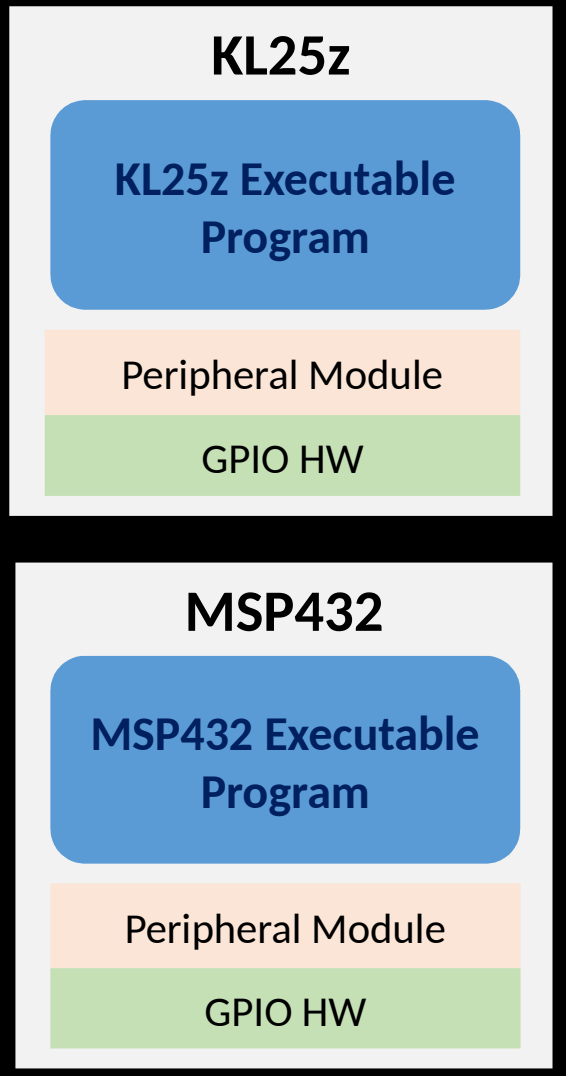
Development Kit



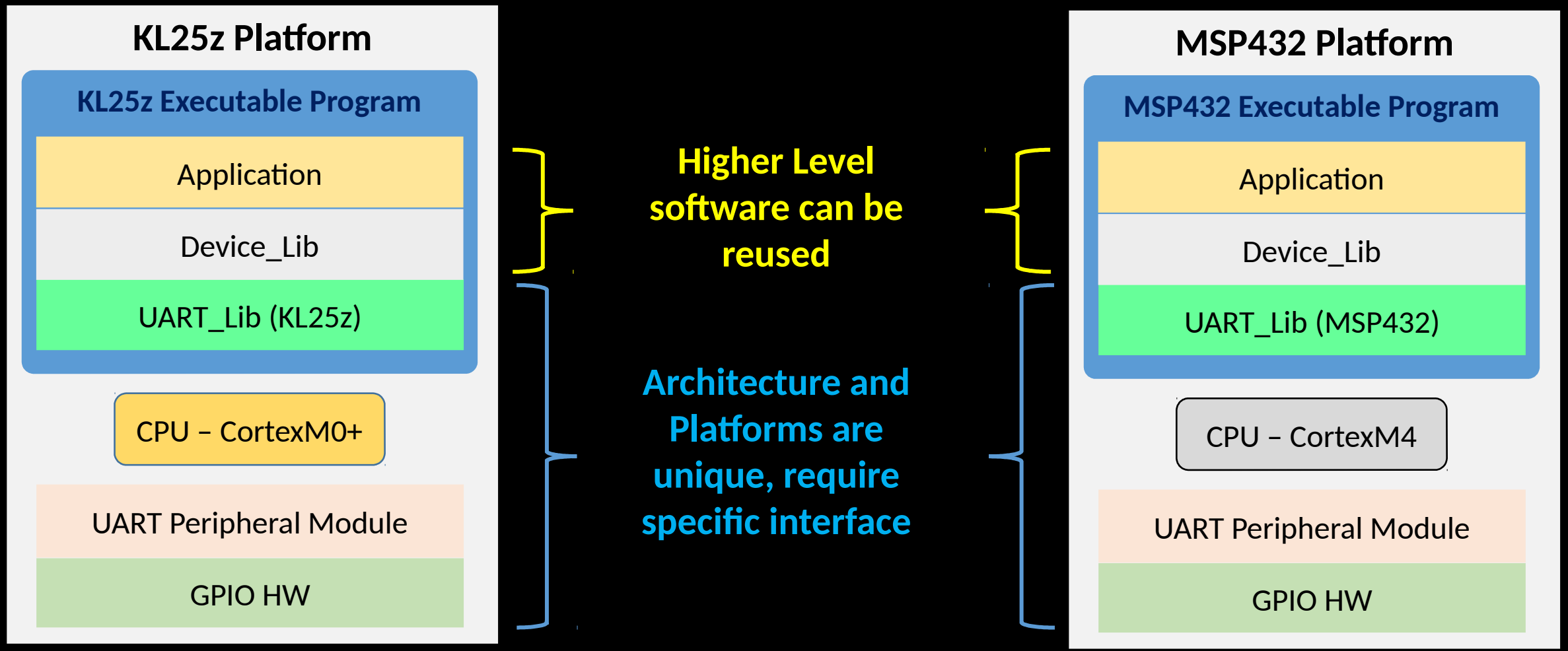
Input File | Generated File | Toolchain Program

Software Independence [S2a]

- Attempt to write as much software with
  - Architecture Independence
  - Platform Independence
- Maximize Software portability and reusability
- Impossible to make everything independent
  - Firmware Layers still interact with hardware
  - Assembly is Architecture Dependent



Software Independence [S3]



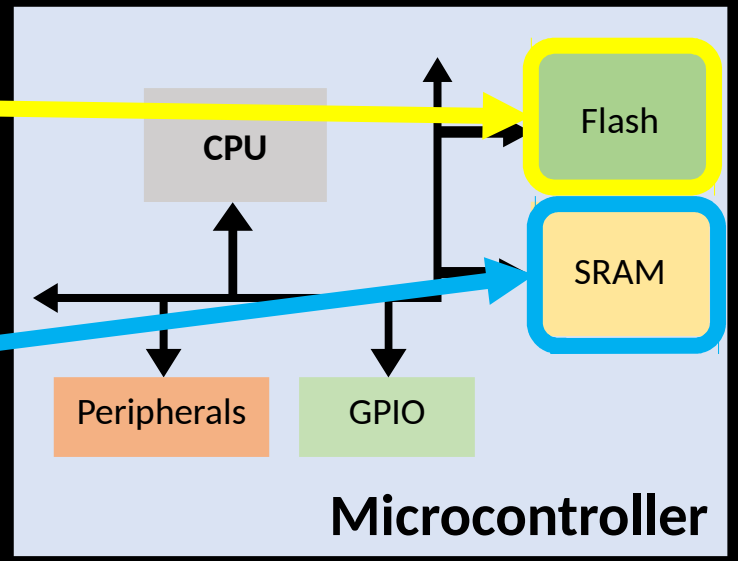
Platform Dependence [S4]

```
MEMORY
{
   MAIN  (RX) : origin = 0x00000000, length = 0x00040000
   DATA  (RW) : origin = 0x20000000, length = 0x00010000
}
SECTIONS
{
   .intvecs :   > 0x00000000
   .text :   > MAIN
   .const :   > MAIN
   .cinit :   > MAIN
   .pinit :   > MAIN
   .data :   > DATA
   .bss :   > DATA
   .heap :   > DATA
   .stack :   > DATA (HIGH)
}
```

Linker Files have Platform Dependence

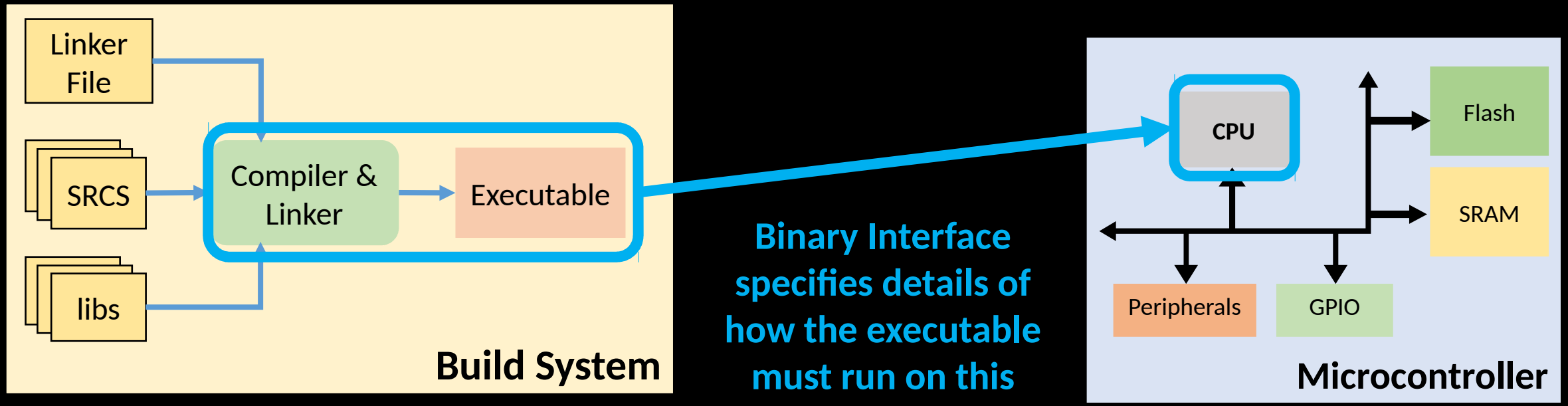
Code Sub-Segments

Data Sub-Segments



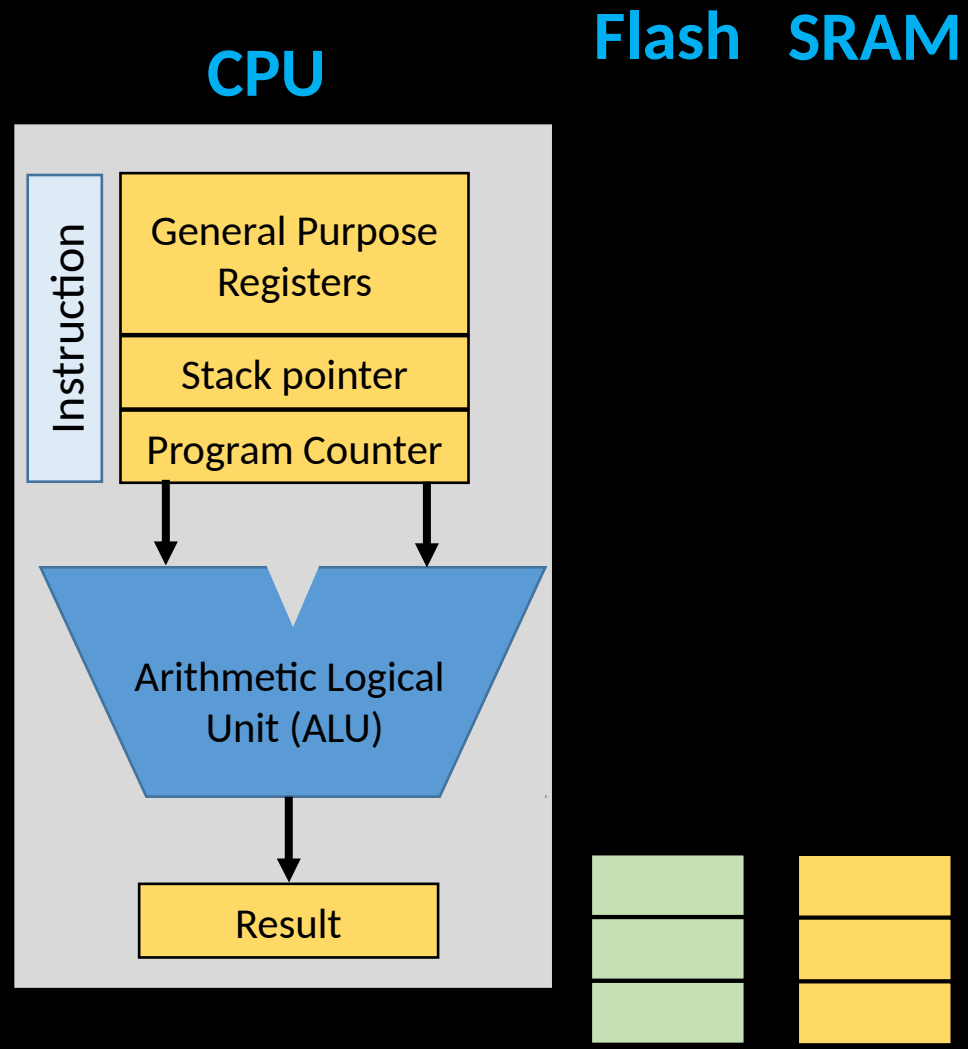
Binary Interface [S5a]

- Compiler and Executable needs to know details on how the architecture should be used at compile time



Binary Interfaces [S5b]

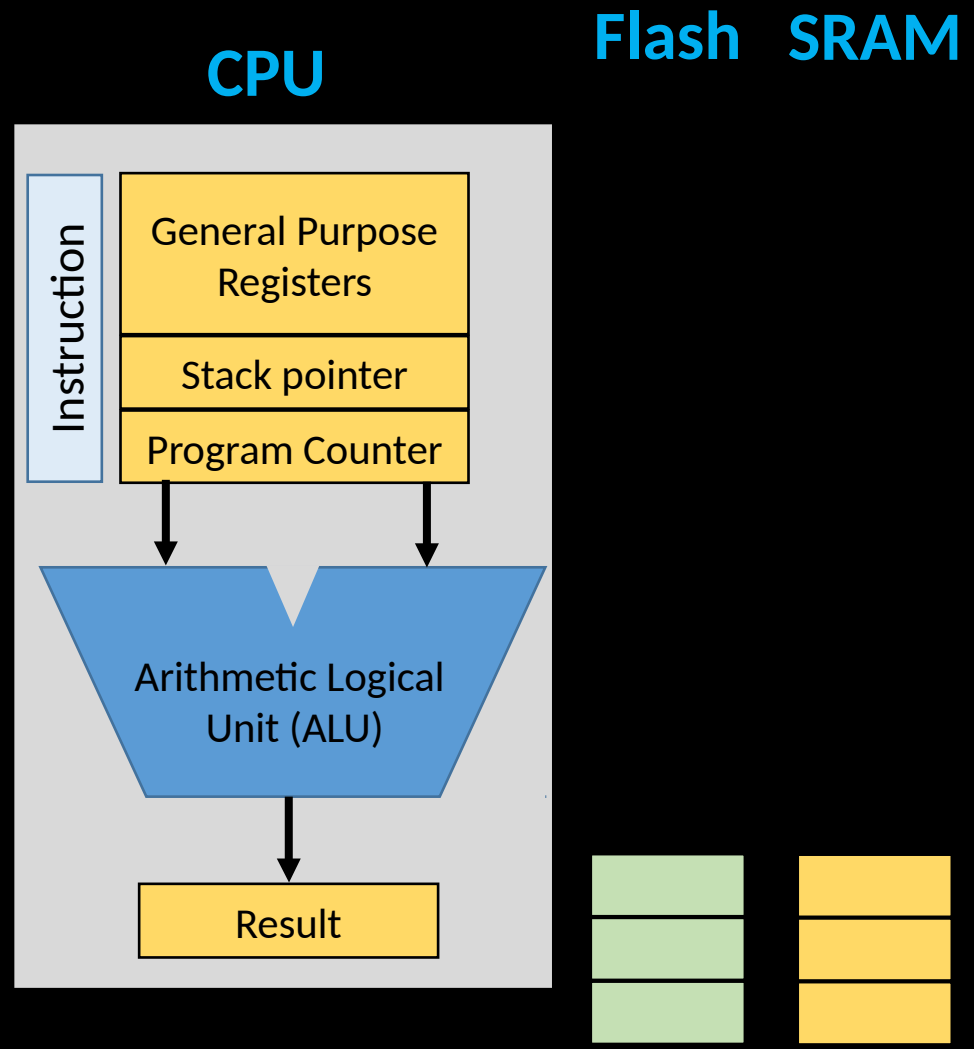
- **Embedded Application Binary Interface (EABI)** – Provides details on how a binary must be compiled and interfaced with platform components
  - Register Use / Word Size
  - Code/Data Storage Requirements
  - Addressing Modes
  - Calling Conventions
  - Helper Functions & Libraries



Binary Interfaces [S6]

- **Embedded Application Binary Interface (EABI)** – Provides details on how a binary must be compiled and interfaced with platform components

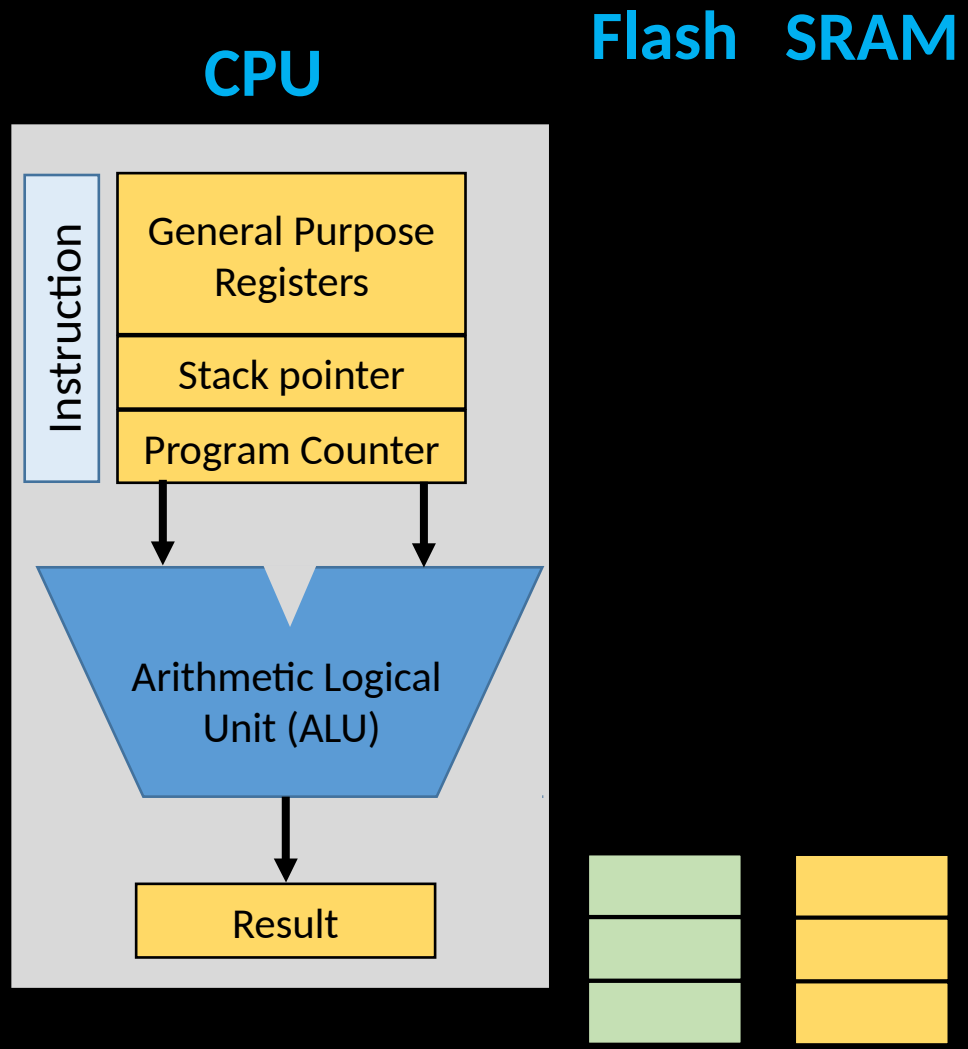
- Registers
  - How many
  - What is the intended uses
- Word Size
  - The operand size of Instruction Set Architecture (ISA)



Binary Interfaces [S7]

- **Embedded Application Binary Interface (EABI)** – Provides details on how a binary must be compiled and interfaced with platform components

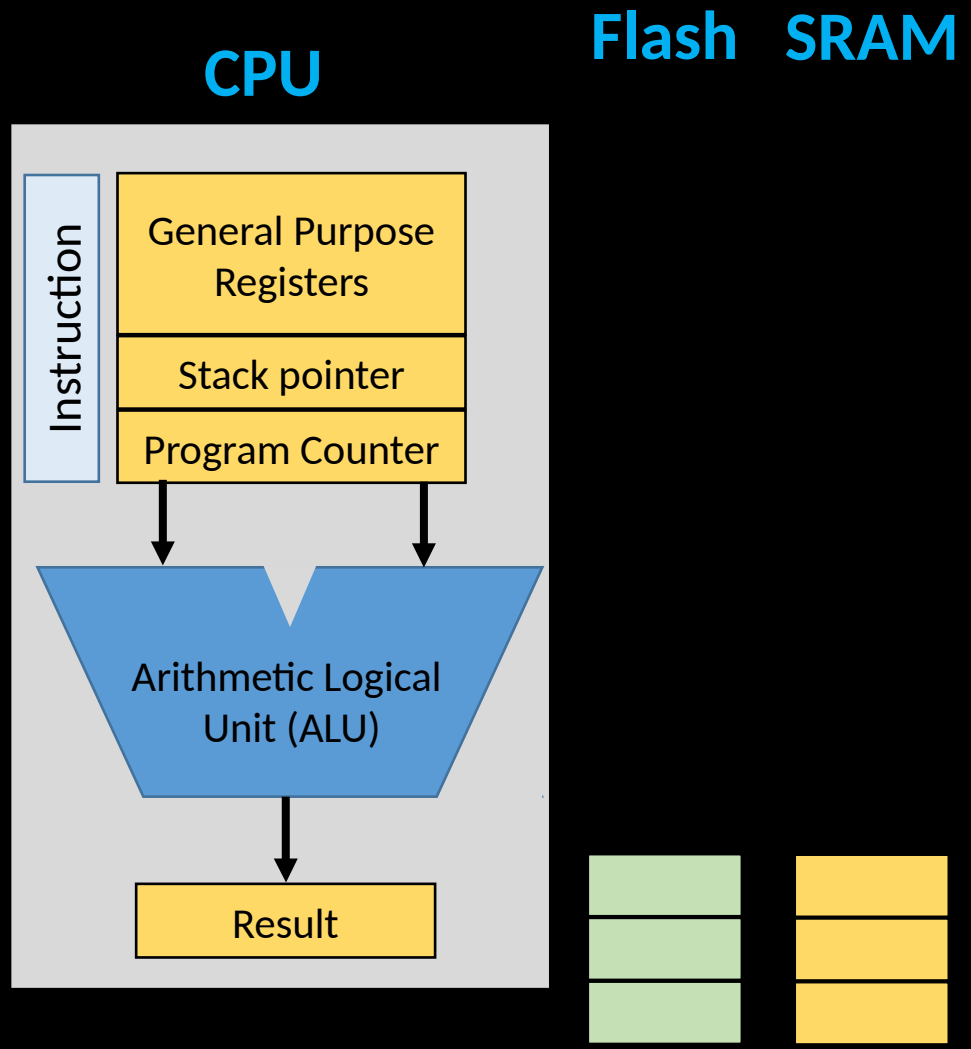
- Program Code & Program Data
  - How large is the instruction?
  - How are they oriented in memory?
  - How large are the data types?
  - How are they aligned?



Binary Interfaces [S8]

- **Embedded Application Binary Interface (EABI)** – Provides details on how a binary must be compiled and interfaced with platform components

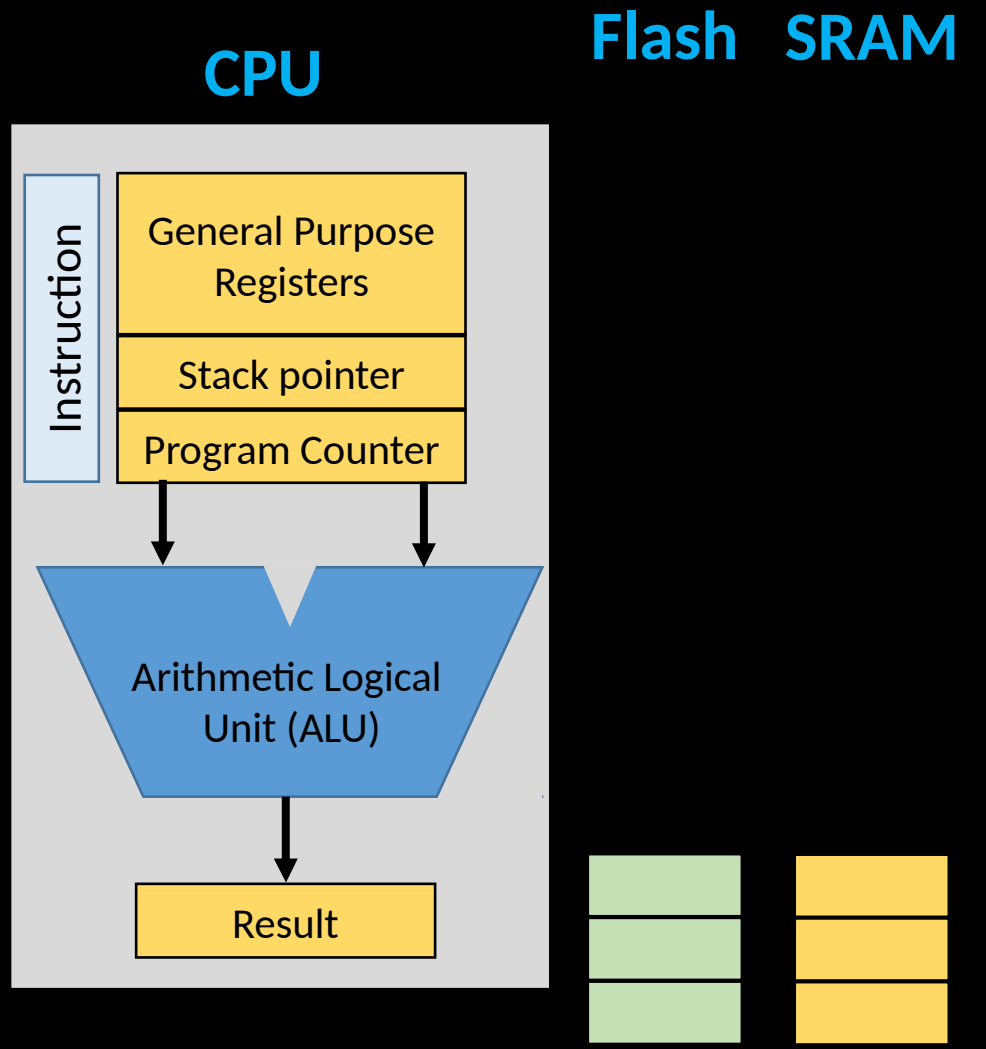
- Addressing Modes
  - Register Addressing
  - Memory Direct Addressing
  - Indirect Addressing
  - Indirect Addressing with Offsets
  - Etc.



Binary Interfaces [S9]

- **Embedded Application Binary Interface (EABI)** – Provides details on how a binary must be compiled and interfaced with platform components

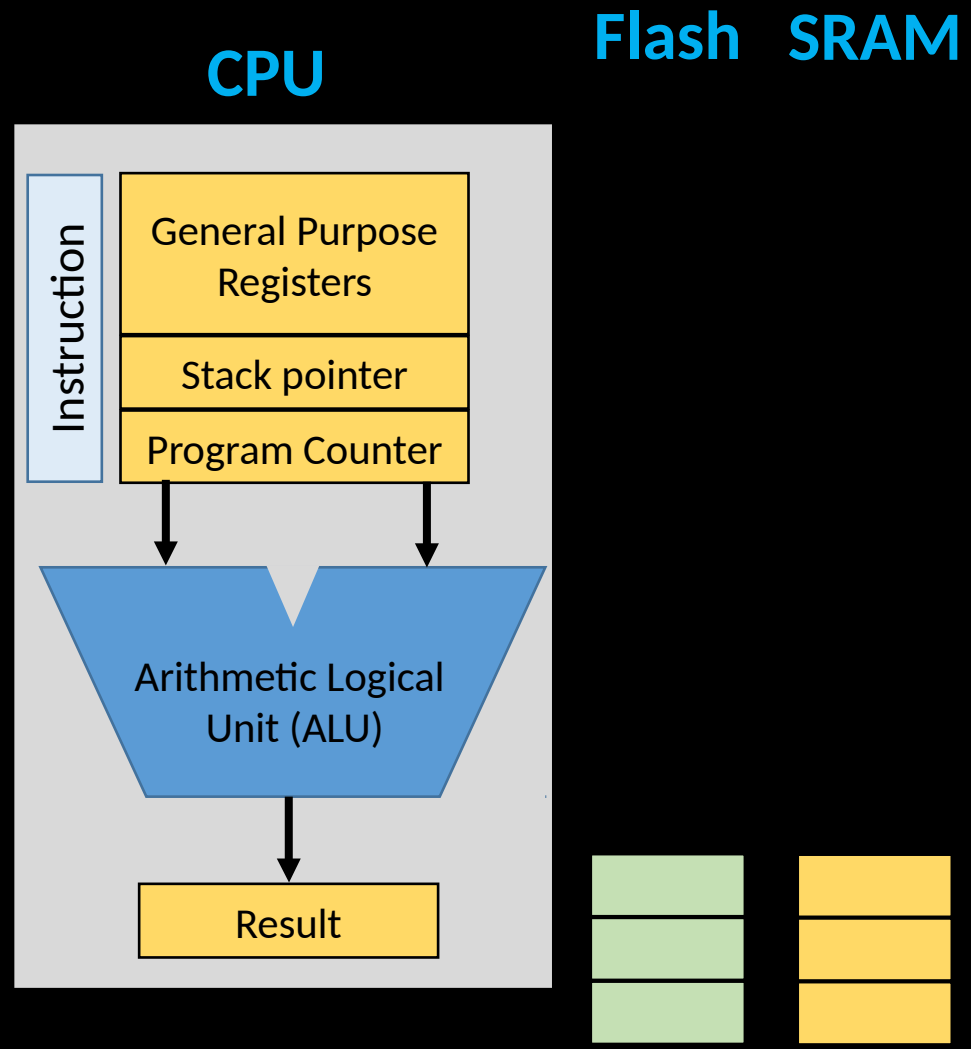
- Calling Convention
  - How stack is managed
  - How routines are called
  - How data is passed in
  - How data is returned
  - What state is saved



Binary Interfaces [S10]

- **Embedded Application Binary Interface (EABI)** – Provides details on how a binary must be compiled and interfaced with platform components

- Helper Functions & Libraries
  - More complex software operations
  - Floating Point Math
  - C-standard Library



Module Outcomes [S11]

At the end of this Module students will be able to...

- Create C-Pointers to read and write to different parts of the ARM Microcontroller Memory Map
- Describe relationship between ARM architecture and C-Programming memory interactions
- Analyze register definitions and design register interface files