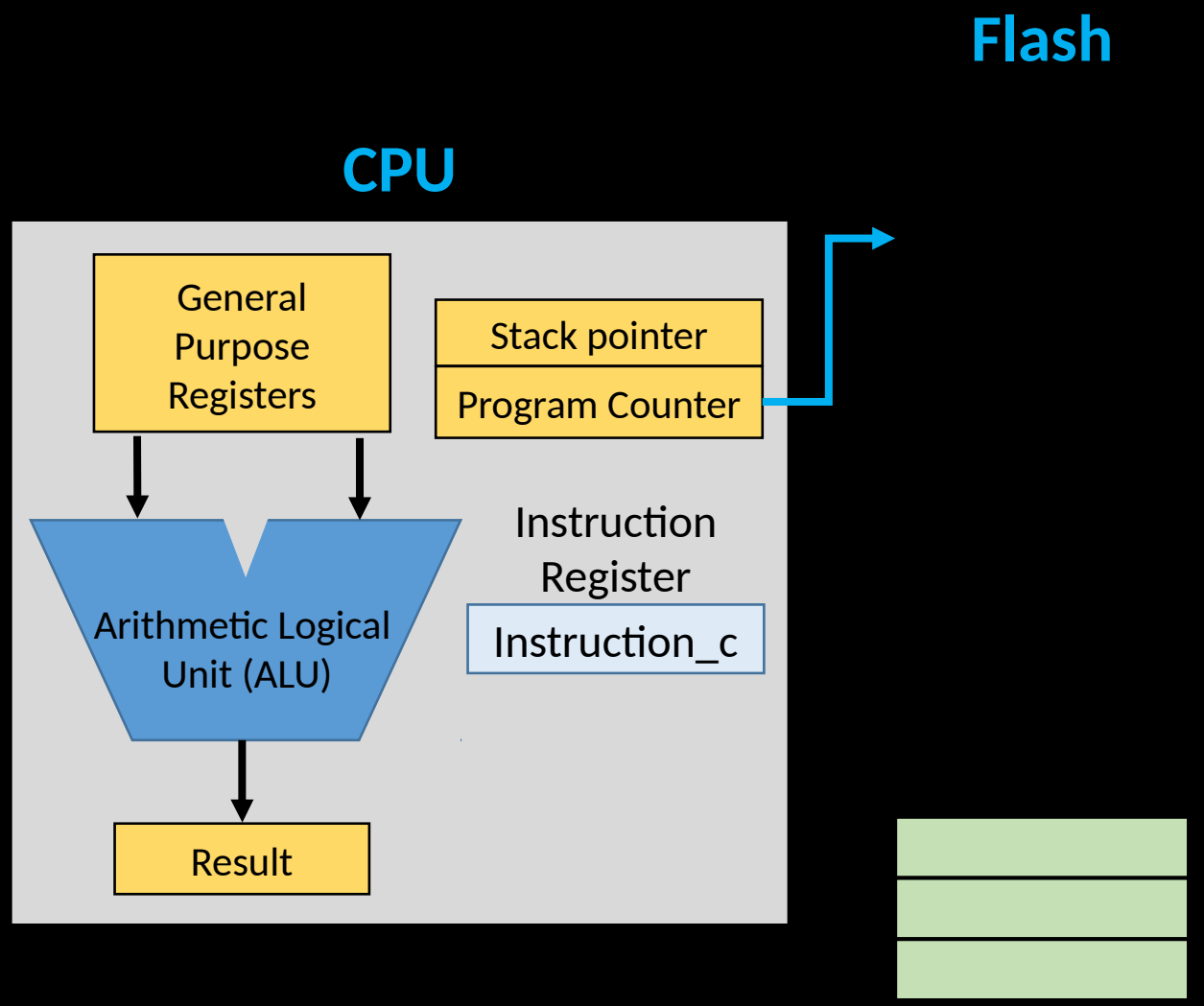
Word Size and Data Types

Embedded Software Essentials

C2M1V2

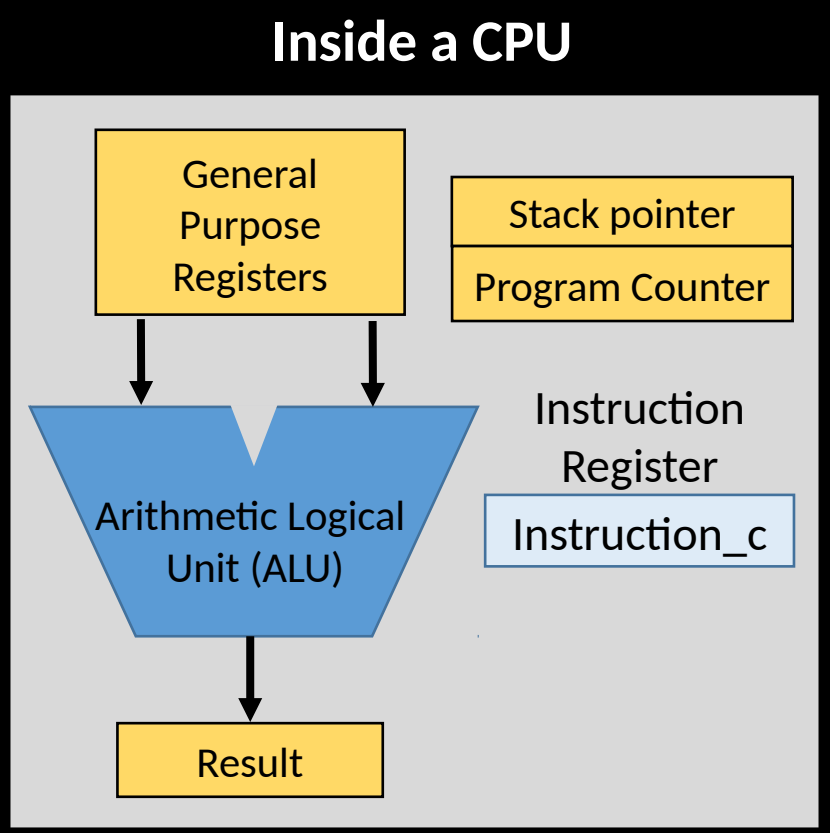
Architecture [S1]

- Architecture designed to implement assembly instructions
- Complex Instruction Set Computer (CISC)
- Reduced Instruction Set Computer (RISC)
- Advanced RISC Machine (ARM)



Units of Operation [S2]

- Instruction – Fundamental unit of work or operation
  - Arithmetic
  - Logical
  - Program Flow Control
  - Load/Store



- Word – fundamental operand size for each operation

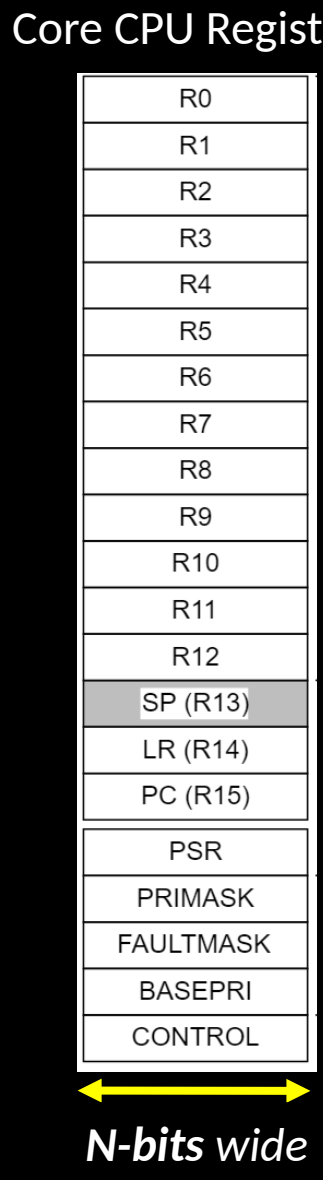
Example: 32-bit Machines are built to do 32-bit math most optimally

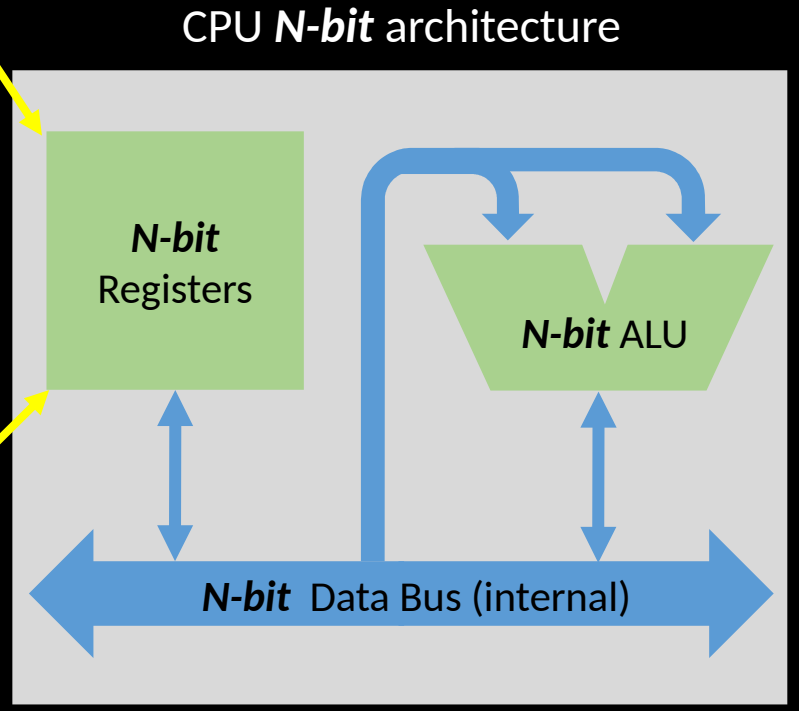
Units of Operation [S3]

- General Purpose Registers in CPU will be size of the Word

  ARM 32-bit Word = 32-bit registers

- Cortex-M Processors has General and Special Purpose CPU Core registers
  - R0-R12 General Purpose
  - R13-R15 Reserved Role
  - Program Status Registers
  - Exception Mask Registers
  - Control Register



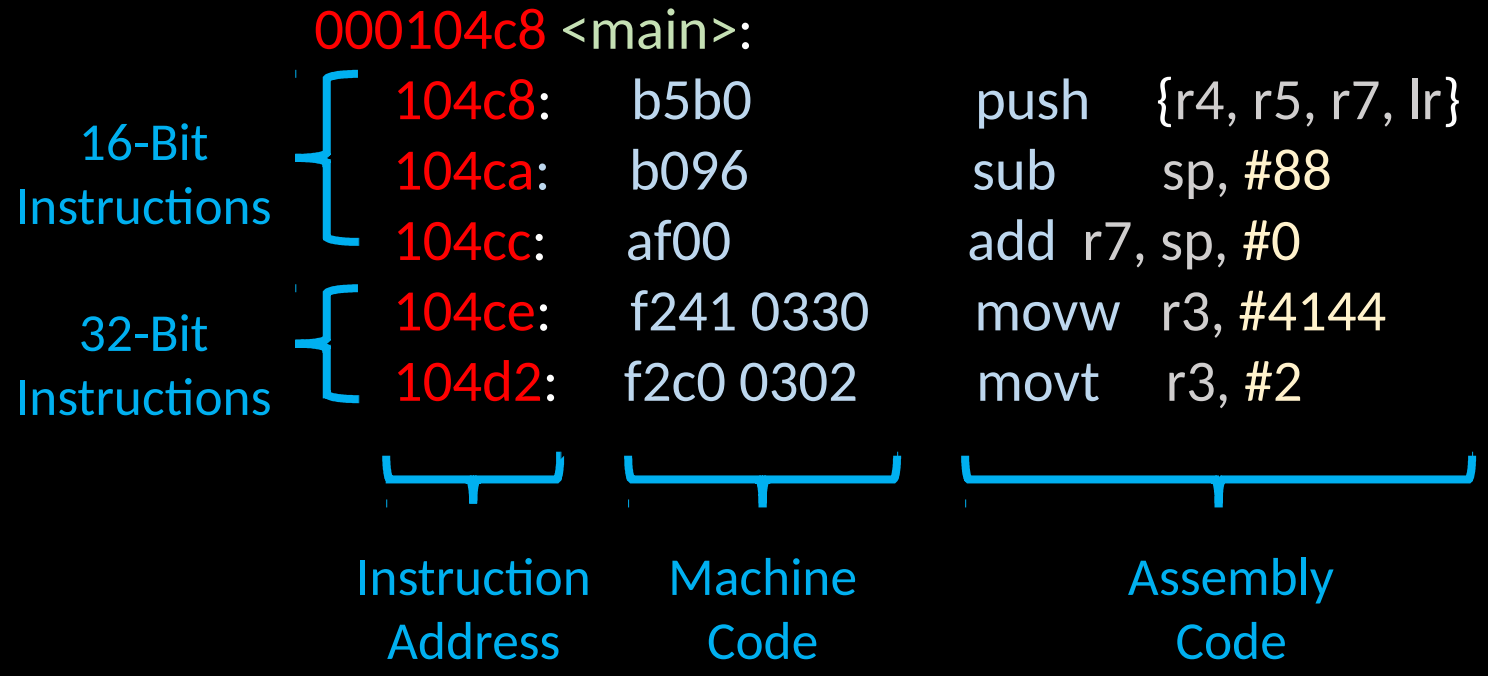


CPU operand size = data size of registers

Instruction Sizes [S4]

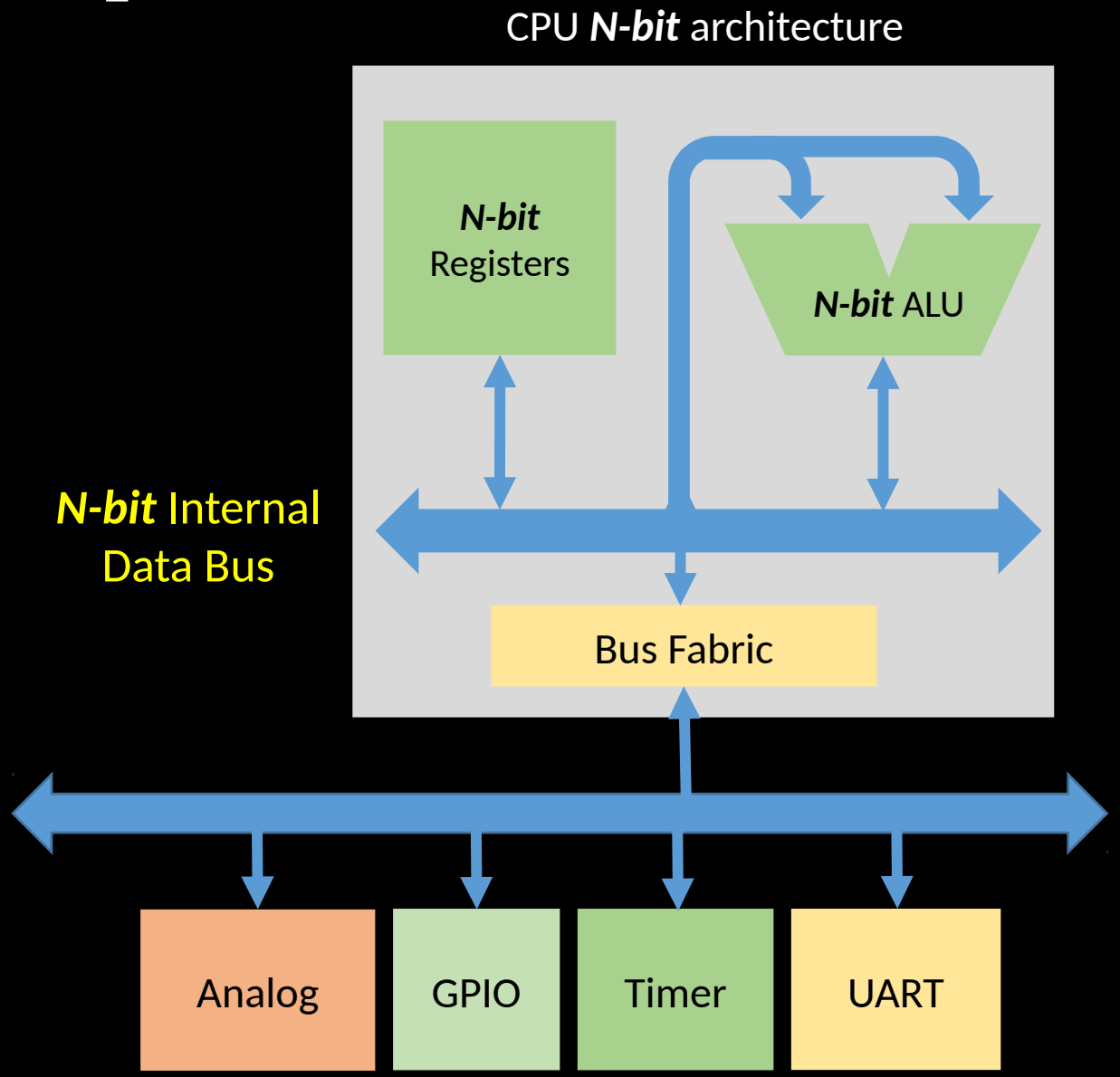
- Instruction size can vary
  - ARM Instruction Set ARMv6-M ⎵ 16-Bit and 32-Bit
  - Thumb-2 Instruction Set ⎵ 16-Bit

Example Output with Machine Code and Assembly Code



Units of Operation [S5]

- There are a lot of Busses in a Microcontroller
  - Internal System Busses
    - Example: ARM AHB
  - External Peripheral Busses
    - Example: ARM APB
- Bus is at least the size of the instruction



C-Programming Types [S6]

- Sizes of C data types are ambiguous and vary between architectures
- C-Standard Specifies a minimum each variable can be

Type	Size	Value Range (**min**)
signed char int8_t	At least 8-bits	$[-2^7, +2^7 - 1]$
unsigned char uint8_t	At least 8-bits	$[0, +2^8 - 1]$
signed short int int16_t	At least 16-bits	$[-2^{15}, +2^{15} - 1]$
unsigned short int uint16_t	At least 16-bits	$[0, +2^{16} - 1]$
signed long int int32_t	At least 32-bits	$[-2^{15}, +2^{15} - 1]$
unsigned long int uint32_t	At least 32-bits	$[0, +2^{32} - 1]$

Standard Integer Sizes[S7]

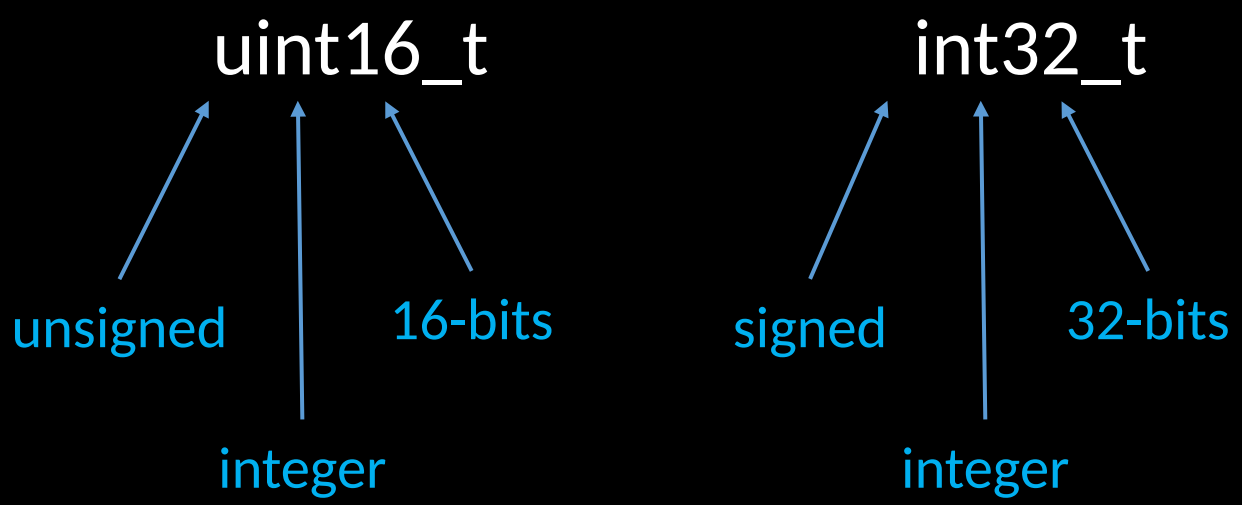
- Variable length Types can cause portability issues
- Explicitly defined types that specify storage and sign
  - Provide exact size, fast size, and minimum size
- Defined in the stdint.h header file

int8_t	→	Exactly 8-bits
int16_t	→	Exactly 16-bits
int32_t	→	Exactly 32-bits
int64_t	→	Exactly 64-bits
uint8_t	→	Exactly 8-bits
uint16_t	→	Exactly 16-bits
uint32_t	→	Exactly 32-bits
uint64_t	→	Exactly 64-bits

Standard Integer Sizes[S8]

- Explicitly defined types that specify exact size and sign
  - U = Unsigned
  - Int = Integer type
  - _t = type

```
uint8_t   var1;
int32_t   var3 = -100;
```




```
#ifndef __STDINT_H__
#define __STDINT_H__

/* 8-bit signed/unsigned Integers */
typedef signed     char int8_t;
typedef unsigned char uint8_t;

/* 16-bit signed/unsigned Integers */
typedef signed     short int int16_t;
typedef unsigned short int uint16_t;

/* 32-bit signed/unsigned Integers */
typedef signed     long int int32_t;
typedef unsigned long int uint32_t;

/* 64-bit signed/unsigned Integers */
typedef signed     long long int int64_t;
typedef unsigned long long int uint64_t

#endif /* __STDINT_H__ */
```

Standard Integer Sizes [S9]

- You need a data size that allows for the fastest access while having at least N bits ▭ Fast Types
  - Typically rounds up to word size
    - Most optimum size for operations

- You need the smallest data size that has a minimum size N bits ▭ Least Types

int_fast8_t	int_least8_t
int_fast16_t	int_least16_t
int_fast32_t	int_least32_t
int_fast64_t	int_least64_t
uint_fast8_t	uint_least8_t
uint_fast16_t	uint_least16_t
uint_fast32_t	uint_least32_t
uint_fast64_t	uint_least64_t

```
int_fast8_t      var1;
uint_least16_t var2 = 12;
```

Typedef [S10]

- Typedef keyword allows programmer to create own types (like an alias)
  - Can apply to standard types or derived types

```
typedef enum Color {
    COLOR_BLUE = 0,
    COLOR_RED = 1,
    COLOR_GREEN = 2,
} Color_t;
```

```
typedef struct Data {
   int32_t  temperature;
   unt32_t date;
   unt32_t time;
} Data_t;
```

```
#ifndef   __STDINT_H__
#define   __STDINT_H__

/* 8-bit signed/unsigned Integers */
typedef signed     char int8_t;
typedef unsigned char uint8_t;

/* 16-bit signed/unsigned Integers */
typedef signed     short int int16_t;
typedef unsigned short int uint16_t;

/* 32-bit signed/unsigned Integers */
typedef signed     long int int32_t;
typedef unsigned long int uint32_t;

/* 64-bit signed/unsigned Integers */
typedef signed     long long int int64_t;
typedef unsigned long long int uint64_t

#endif /* __STDINT_H__ */
```