Pomona College

Computer Science

Spring 2025: CSCI 181RT

Real-Time Systems in the Real World

Lecture 4

Thursday, September 5, 2024
Edmunds Hall 105
2:45 PM - 4:00 PM

Professor Jennifer DesCombes

Agenda

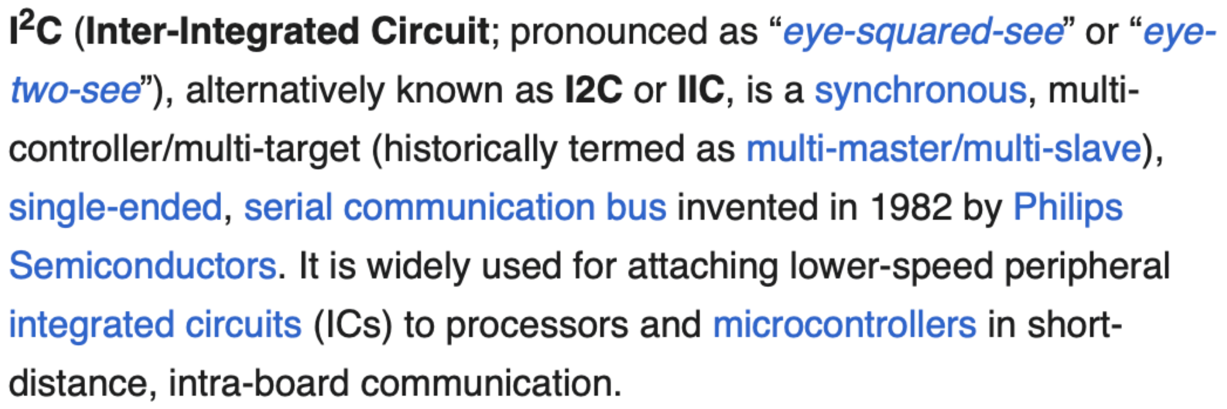
- Go Backs
- Survey Results
- Discussion on Reading
- Discussion on Lab
- A Bit More On Simple Real-Time Systems
- Computer Memory Utilization
- Initial Investigation of Interrupts
- Assignment
- Look Ahead
- Action Items

Announcements

- Mentor Session
    - Neil Chulani
    - email njcf2022@mymail.pomona.edu
    - Time: 1:00 - 3:00 - Every Sunday
    - Where: Edmonds 105 (here)
    - Snacks Provided!

Go Backs

- General?
- Action Item Status
  - AI250128-1: Mitchell awarded 1% for missed update on *“Fall 2024”* in Lecture Charts. - OK to Close?
  - AI250128-2: Cameron awarded 1% for missed update on *“2024”* in multiple Lecture Charts. - OK to Close?
  - AI250128-3: David awarded 1% for missed update on *“Jan 21”* in Lecture Charts. - OK to Close?
  - AI250128-4: Define I2C. - OK to Close?



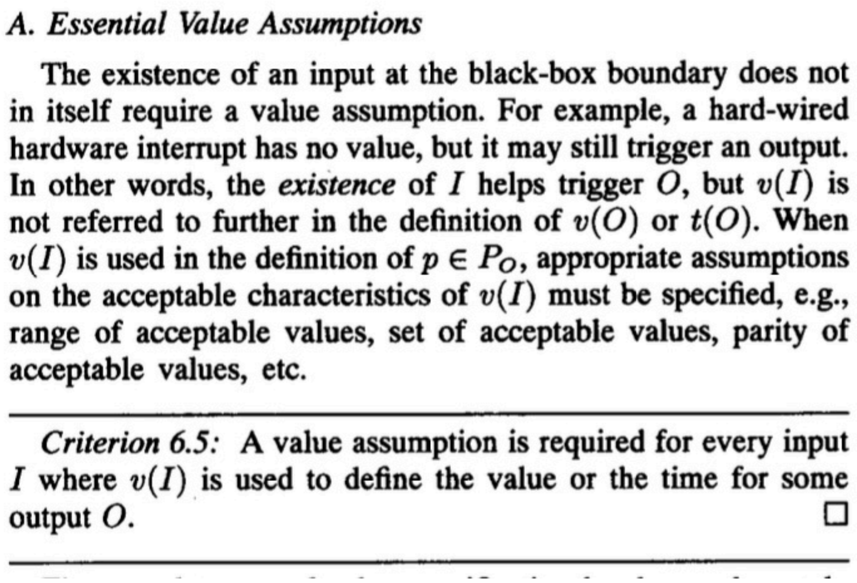
I^{2}C (**Inter-Integrated Circuit**; pronounced as “*eye-squared-see*” or “*eye-two-see*”), alternatively known as **I2C** or **IIC**, is a synchronous, multi-controller/multi-target (historically termed as multi-master/multi-slave), single-ended, serial communication bus invented in 1982 by Philips Semiconductors. It is widely used for attaching lower-speed peripheral integrated circuits (ICs) to processors and microcontrollers in short-distance, intra-board communication.

Go Backs

- Action Item Status (continued)
  - AI250128-5: Mitchell awarded 1% for error on *“Volume -“* in Lecture Charts. - OK to Close?
  - AI250128-6: Provide Email Sort String for Reading Questions. - OK to Close?
  - AI250128-7: Send out Lecture Charts to students that added the course. - OK to Close?
  - AI250129-1: Send out syllabus to students that added the course. - OK to Close?
  - AI250129-2: Alisha awarded 1% for spelling error on *Compiler* in Lab Charts. - OK to Close?

Discussion on Reading

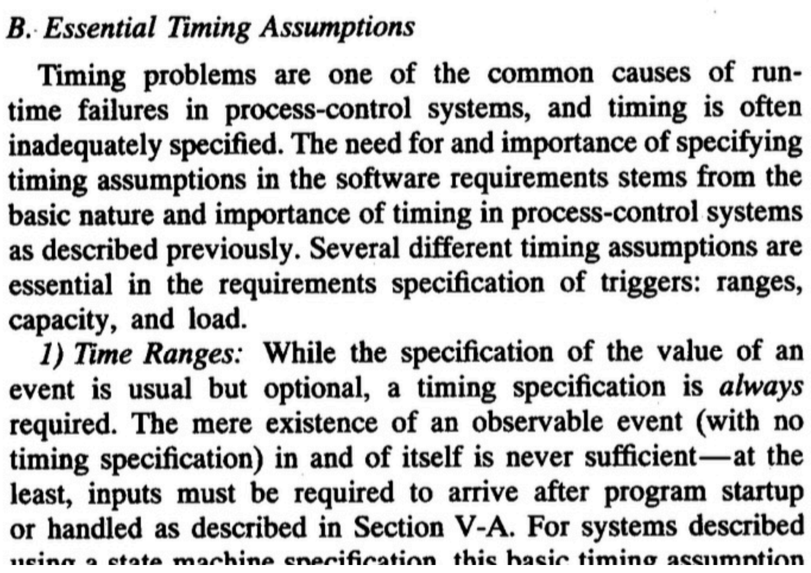
- K & R
  - General Questions?
- Journal Article - Software Requirements Analysis for Real-Time Process-Control Systems
  - Complexity?



A. Essential Value Assumptions

The existence of an input at the black-box boundary does not in itself require a value assumption. For example, a hard-wired hardware interrupt has no value, but it may still trigger an output. In other words, the *existence* of I helps trigger O, but $v(I)$ is not referred to further in the definition of $v(O)$ or $t(O)$. When $v(I)$ is used in the definition of $p \in P_O$, appropriate assumptions on the acceptable characteristics of $v(I)$ must be specified, e.g., range of acceptable values, set of acceptable values, parity of acceptable values, etc.

Criterion 6.5: A value assumption is required for every input I where $v(I)$ is used to define the value or the time for some output O. □



B. Essential Timing Assumptions

Timing problems are one of the common causes of run-time failures in process-control systems, and timing is often inadequately specified. The need for and importance of specifying timing assumptions in the software requirements stems from the basic nature and importance of timing in process-control systems as described previously. Several different timing assumptions are essential in the requirements specification of triggers: ranges, capacity, and load.

1) Time Ranges: While the specification of the value of an event is usual but optional, a timing specification is *always* required. The mere existence of an observable event (with no timing specification) in and of itself is never sufficient—at the least, inputs must be required to arrive after program startup or handled as described in Section V-A. For systems described using a state machine specification, this basic timing assumption

Discussion on Lab

- Hello World - Done
- Toolset Download and Install - Mostly Done (confirm when compile code)
- Questions?

A Simple Real-Time System Design
Elevator Music Controller

- Two Types of Logic and Control Approaches
- Combined Data and Control Blocks
  - Logically Simple - Works for Simple Systems
  - Acquire Input Data (button state), Take Action
  - Examine Next Input Data
- Separate Data and Control Blocks
  - Can Support More Complex Logic
  - Acquire All Data
  - Take All Actions Based on All Acquired Data
- Has Impact on Team Development, Simulation, and Testing

NOTE: Date reflects original presentation date.

A Simple Real-Time System Design
Elevator Music Controller

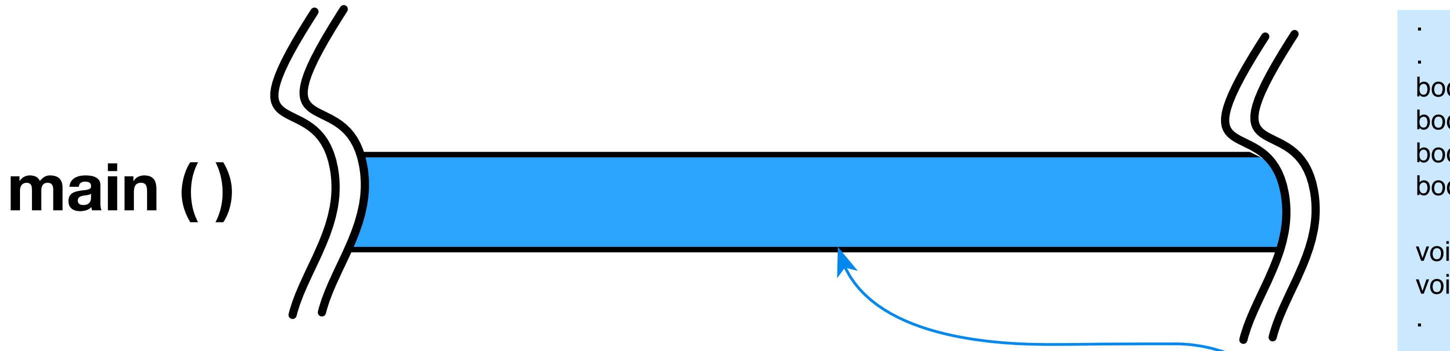
```
.
.
bool button1Pressed;
bool button2Pressed;
bool button3Pressed;
bool button4Pressed;

void readButtons( void );
void processButtons ( void );
.
.
  while(true)
  {
    readButtons( );
    processButtons( );
  }
.
.
```

Declarations of global variables and functions.

Separate Data and Control Blocks (routines, methods, etc.)

A Simple Real-Time System Design
Elevator Music Controller

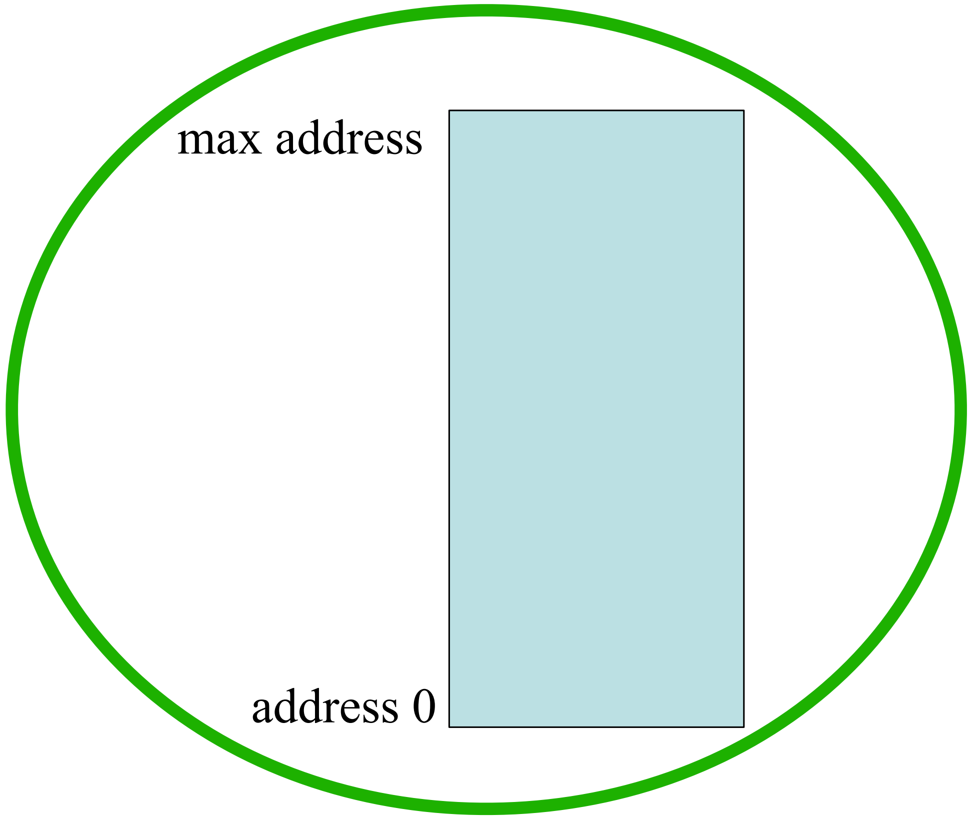


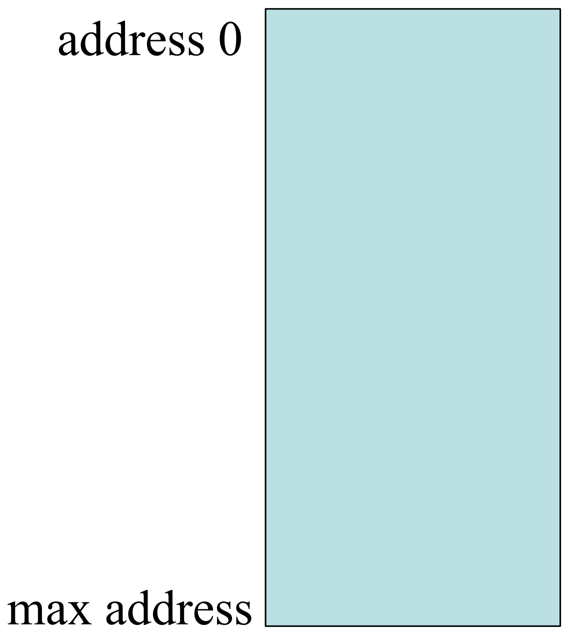
```
.
.
bool button1Pressed;
bool button2Pressed;
bool button3Pressed;
bool button4Pressed;

void readButtons( void );
void processButtons ( void );
.
.
  while(true)
  {
    readButtons( );
    processButtons( );
  }
.
.
```

Computer Memory Utilization

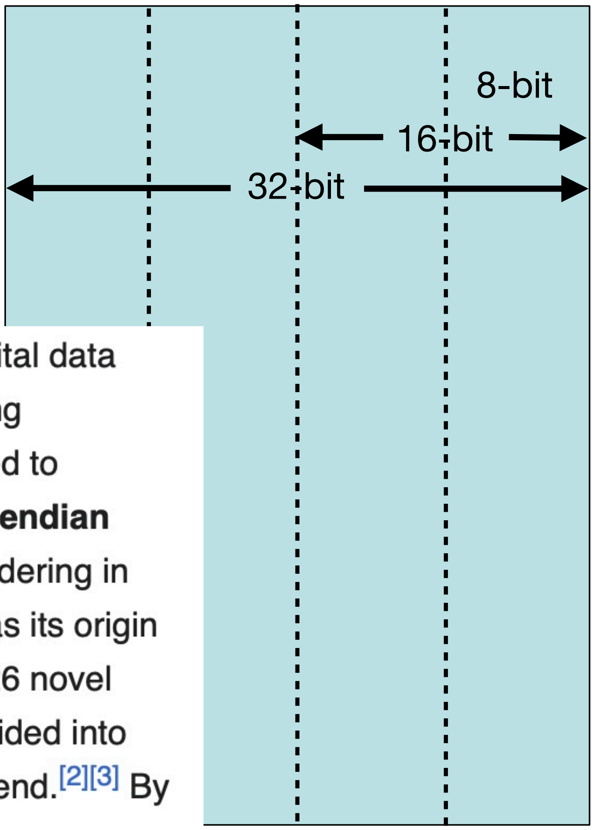
- For This Discussion, Treat Memory as Contiguous & Real
  - Top, Bottom, Upside Down?

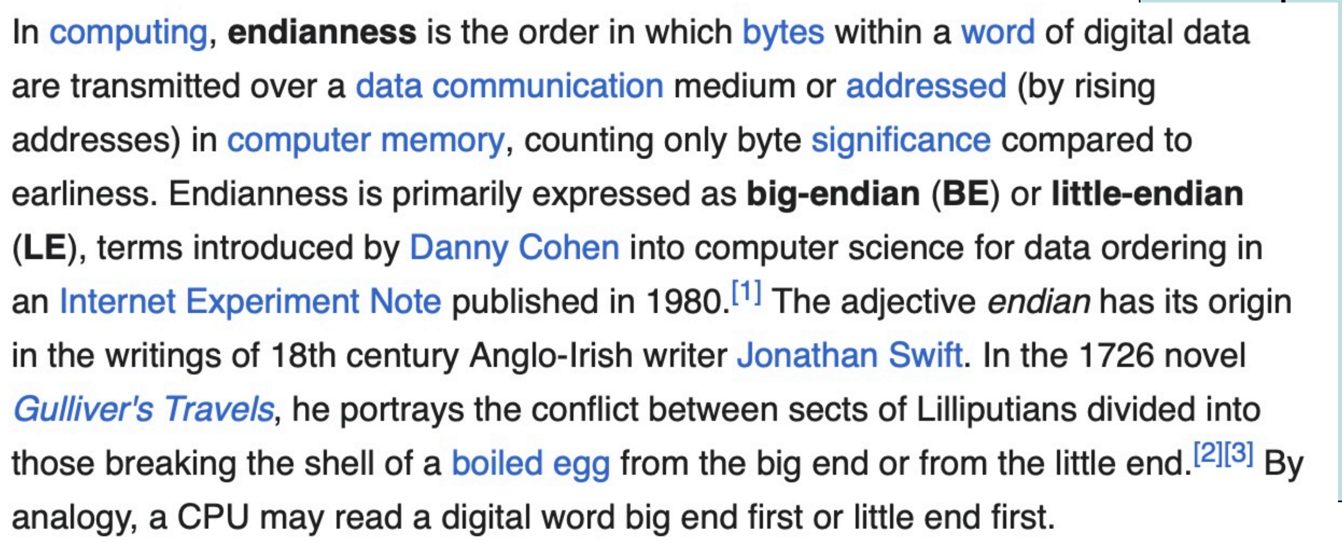




Computer Memory Utilization

- For This Discussion, Treat Memory as Contiguous & Real
  - Organization
    - 8, 16, 32, 64-bit
    - Access
    - Endianness




In computing, **endianness** is the order in which bytes within a word of digital data are transmitted over a data communication medium or addressed (by rising addresses) in computer memory, counting only byte significance compared to earliness. Endianness is primarily expressed as **big-endian** (**BE**) or **little-endian** (**LE**), terms introduced by Danny Cohen into computer science for data ordering in an Internet Experiment Note published in 1980.[1] The adjective *endian* has its origin in the writings of 18th century Anglo-Irish writer Jonathan Swift. In the 1726 novel *Gulliver's Travels*, he portrays the conflict between sects of Lilliputians divided into those breaking the shell of a boiled egg from the big end or from the little end.[2][3] By analogy, a CPU may read a digital word big end first or little end first.

Pomona College

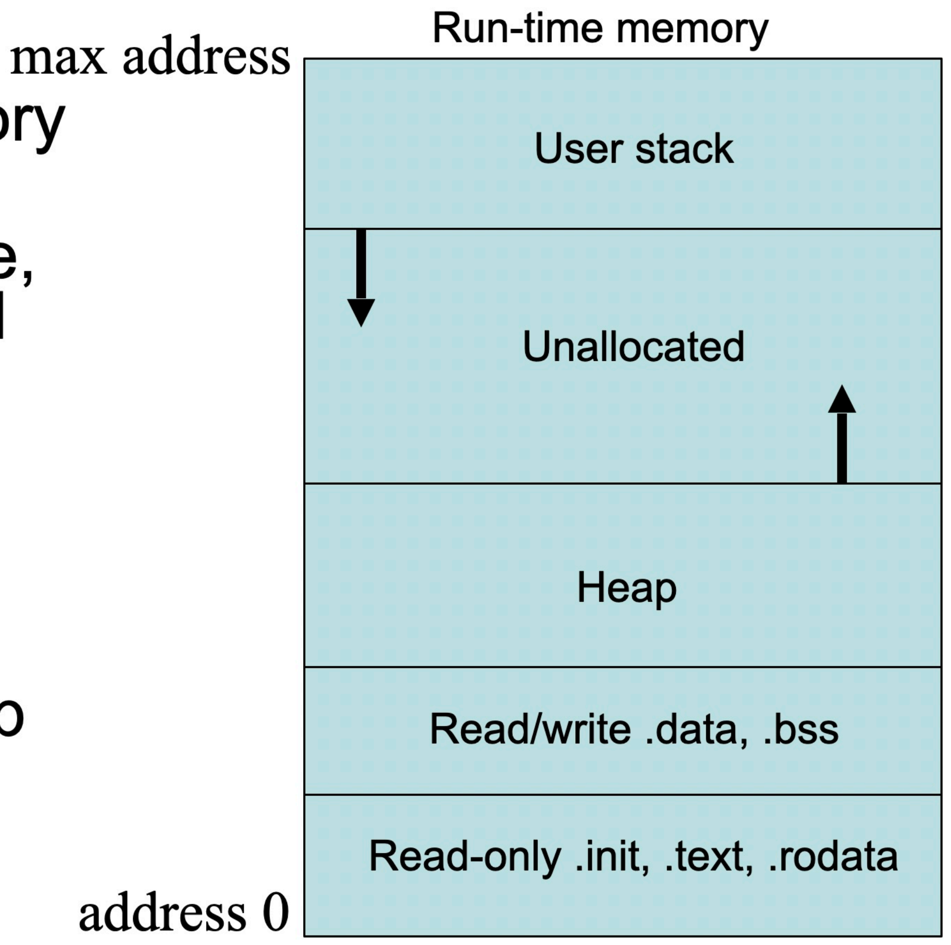
Computer Memory Utilization

- For This Discussion, Treat Memory as Contiguous & Real
  - Program Space (if loaded from storage)
  - Memory Mapped IO
  - Constants
  - Global Variables
  - Heap (dynamic allocation)
  - Stack (dynamic use)

Source: CU Boulder - CSCI 3753 Operating Systems

Stack Behavior

- Run-time memory image
- Essentially code, data, stack, and heap
- Code and data loaded from executable file
- Stack grows downward, heap grows upward



Computer Memory Utilization - Tools

- Stack
  - Push, Pop (Pull)
  - Frame Creation
    - Save Prior Frame Pointer (dedicated variable, register, other)
    - Allocate Space on the Stack
  - Stack Overflow
- Heap
  - Allocate, Deallocate
  - Fragmentation
  - Leaks

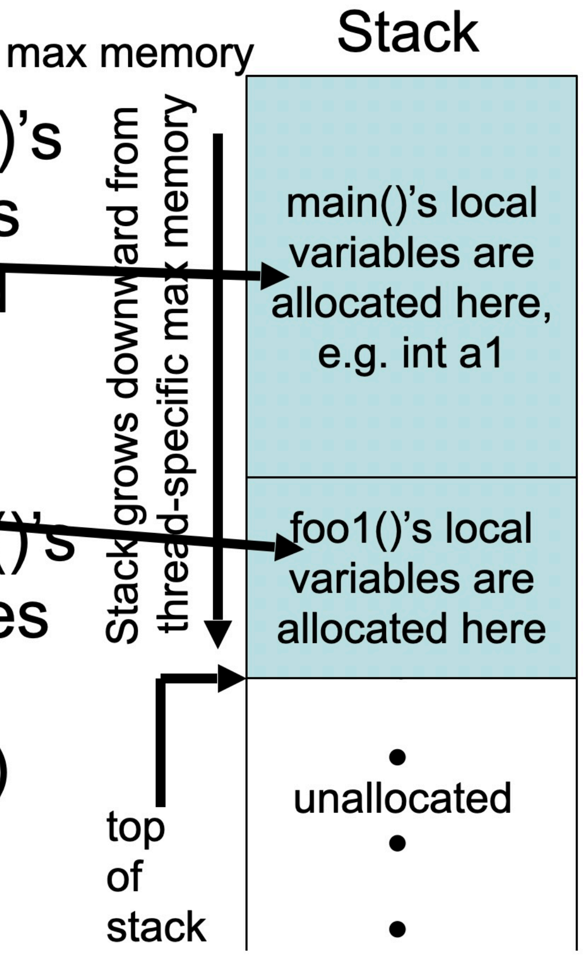
Source: CU Boulder - CSCI 3753 Operating Systems

Relating the Code to the Stack

```
main() {
   int a1;
   ...
   foo1();
   ...
}
```

space for main()'s local variables gets allocated on the stack

space for foo1()'s local variables is allocated below main() on the stack


Stack
max memory
Stack grows downward from thread-specific max memory
main()'s local variables are allocated here, e.g. int a1
foo1()'s local variables are allocated here
unallocated
top of stack

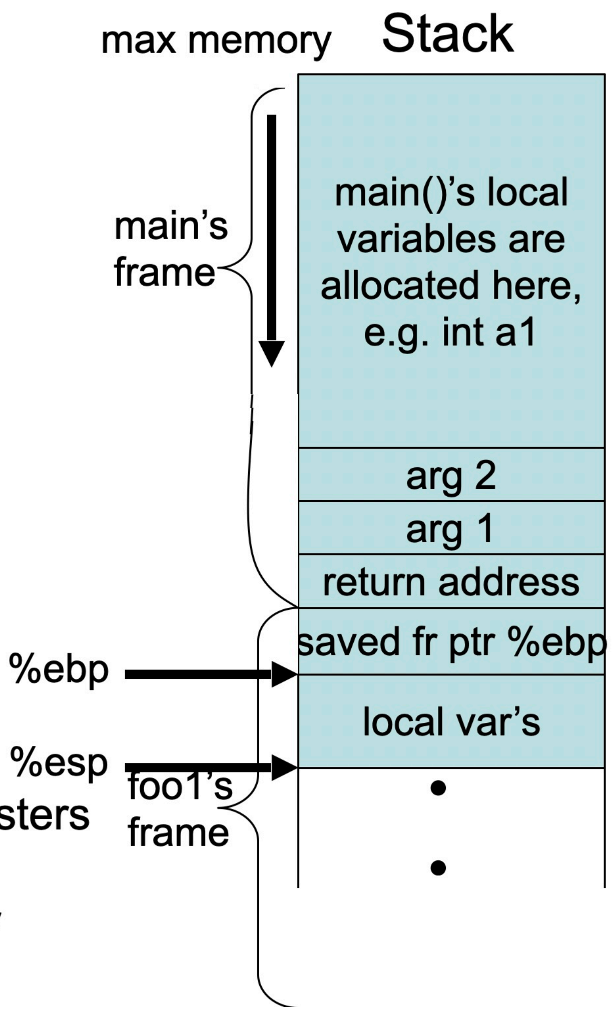
Source: CU Boulder - CSCI 3753 Operating Systems

Entering a Function

```
foo1(int v1, v2) {
PC ──► local var's
       ...
}
```

assembly code:

- foo1 first saves the old frame pointer by pushing it onto the stack: pushl %ebp
- foo1 resets frame ptr to new base (current stack ptr): movl %esp, %ebp
- foo1 saves any callee CPU registers on stack (not shown)
- foo1 allocates local variables by decrementing stack ptr



Initial Investigation of Interrupts

- Complexity - Next Level Up from Simple Polling
- Respond to Internal or External Events
  - Internal - Timers, Fault Conditions
  - External - Logic Inputs, Ports, Input Compare, Input Count
- How Do I Stop?
- How Do I Start?
- Hardware and Toolset Awareness

Pomona College

Assignment - Readings

- Lecture Reading

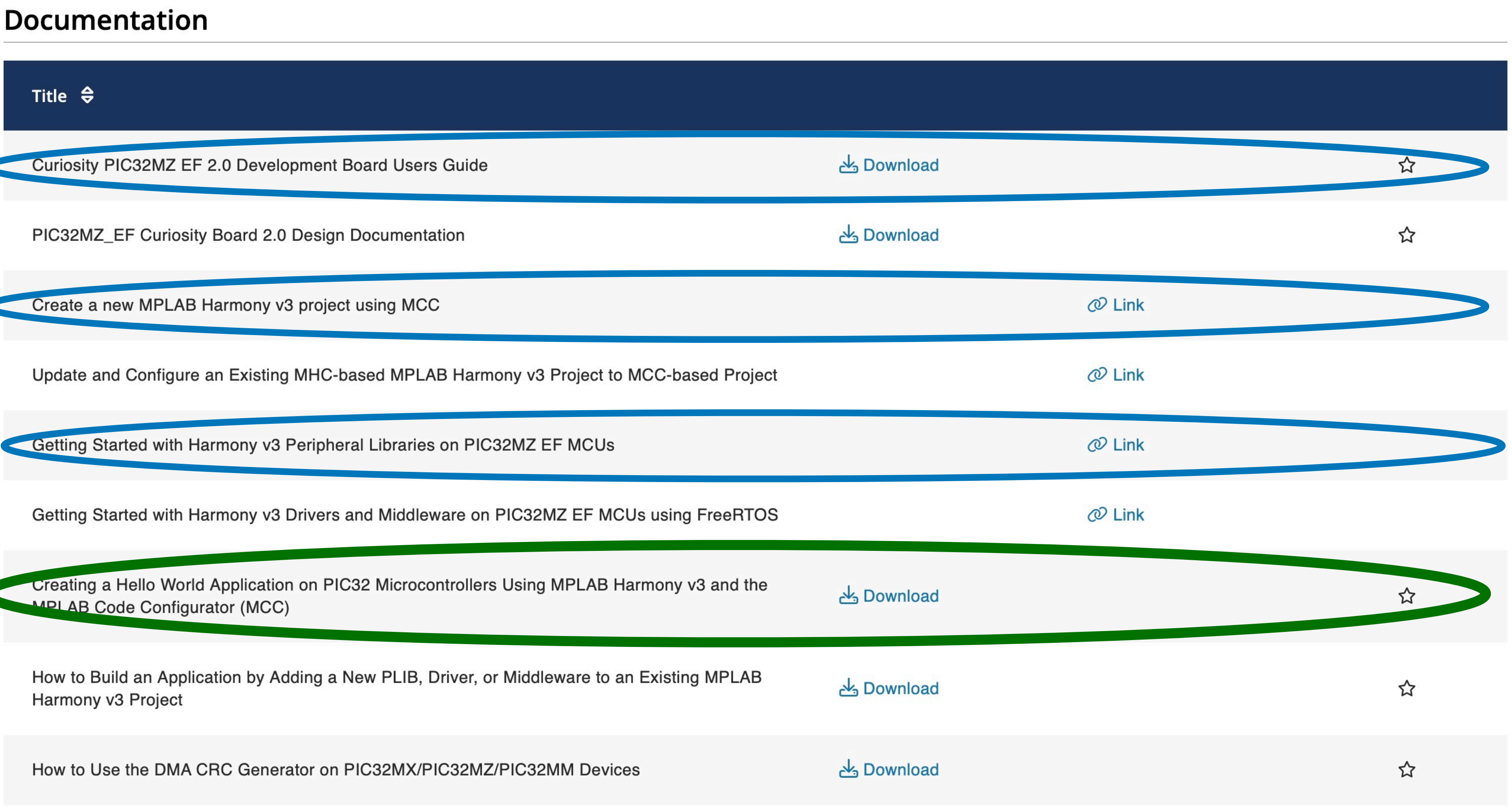
Stacks and Frames Demystified
CU Boulder CSCI 3753 Operating Systems, Spring 2005, Prof. Rick Han
(.pdf provided)

Assignment: Send a Discussion Topic/Question from the CU Presentation to Prof DesCombes by Tuesday 10:00 AM

K&R - Page 67 - 126
Chapter 4: Functions and Program Structure
Chapter 5: Pointers and Arrays

Pomona College

Assignment - Lab Preparation - Virtual Terminal



Documentation

Title		
Curiosity PIC32MZ EF 2.0 Development Board Users Guide	Download	☆
PIC32MZ_EF Curiosity Board 2.0 Design Documentation	Download	☆
Create a new MPLAB Harmony v3 project using MCC	Link	
Update and Configure an Existing MHC-based MPLAB Harmony v3 Project to MCC-based Project	Link	
Getting Started with Harmony v3 Peripheral Libraries on PIC32MZ EF MCUs	Link	
Getting Started with Harmony v3 Drivers and Middleware on PIC32MZ EF MCUs using FreeRTOS	Link	
Creating a Hello World Application on PIC32 Microcontrollers Using MPLAB Harmony v3 and the MPLAB Code Configurator (MCC)	Download	☆
How to Build an Application by Adding a New PLIB, Driver, or Middleware to an Existing MPLAB Harmony v3 Project	Download	☆
How to Use the DMA CRC Generator on PIC32MX/PIC32MZ/PIC32MM Devices	Download	☆

Look Ahead

- Discuss Readings
- More on Interrupt Based Real-time Systems
- Lab Preview

Pomona College

Action Items and Discussion

AI#:	Owner	Slide #	Document	Action