Pomona College

Computer Science

Fall 2024: 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

Pomona College

Agenda

- Go Backs
- Discussion on Reading
- Discussion on Lab
- Computer Memory Utilization
- Initial Investigation of Interrupts
- Assignment
- Look Ahead
- Action Items

Pomona College

Go Backs

- General?
- Lab Start Time Clarification
  - Lab starts nominally at 1:15, promptly at 1:20
- Action Item Status
  - AI240903-1: Send out emails concerning types of preferred tea brands to those who indicated a beverage preference of tea. - OK to Close?

Pomona College

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

- Associates Code Samples Distributed
  - Review Prior to Next Weeks Lab
- Likes? Dislikes?
- Update on Evaluation Boards

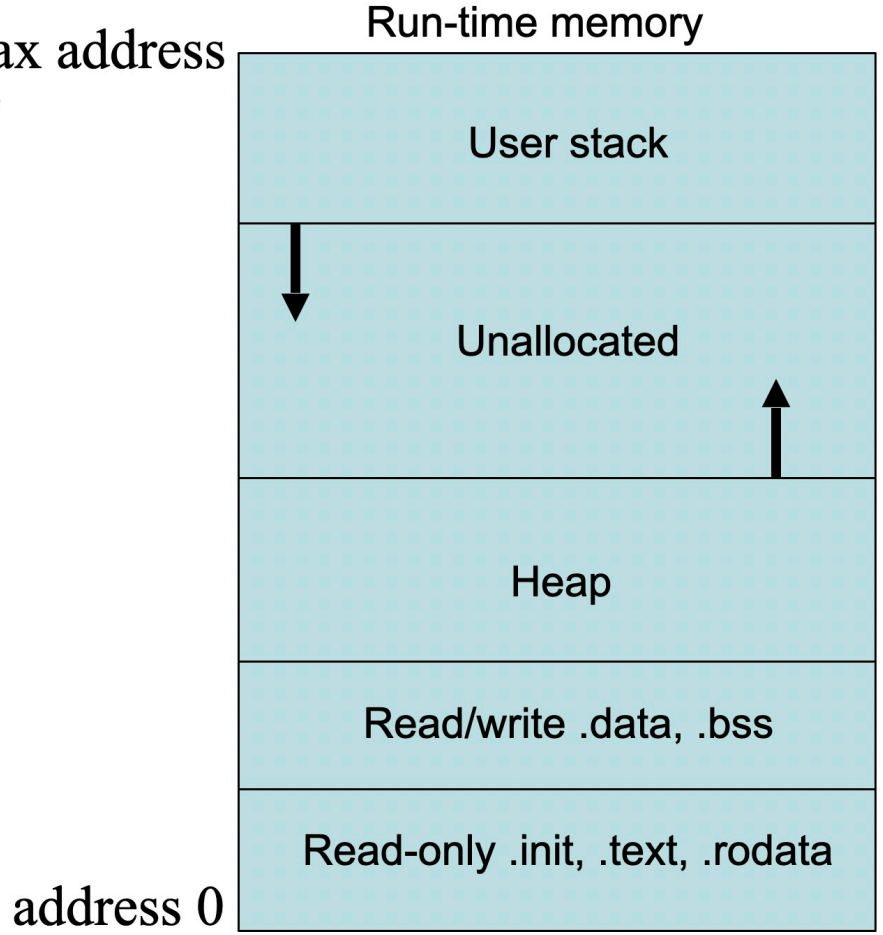
Computer Memory Utilization

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

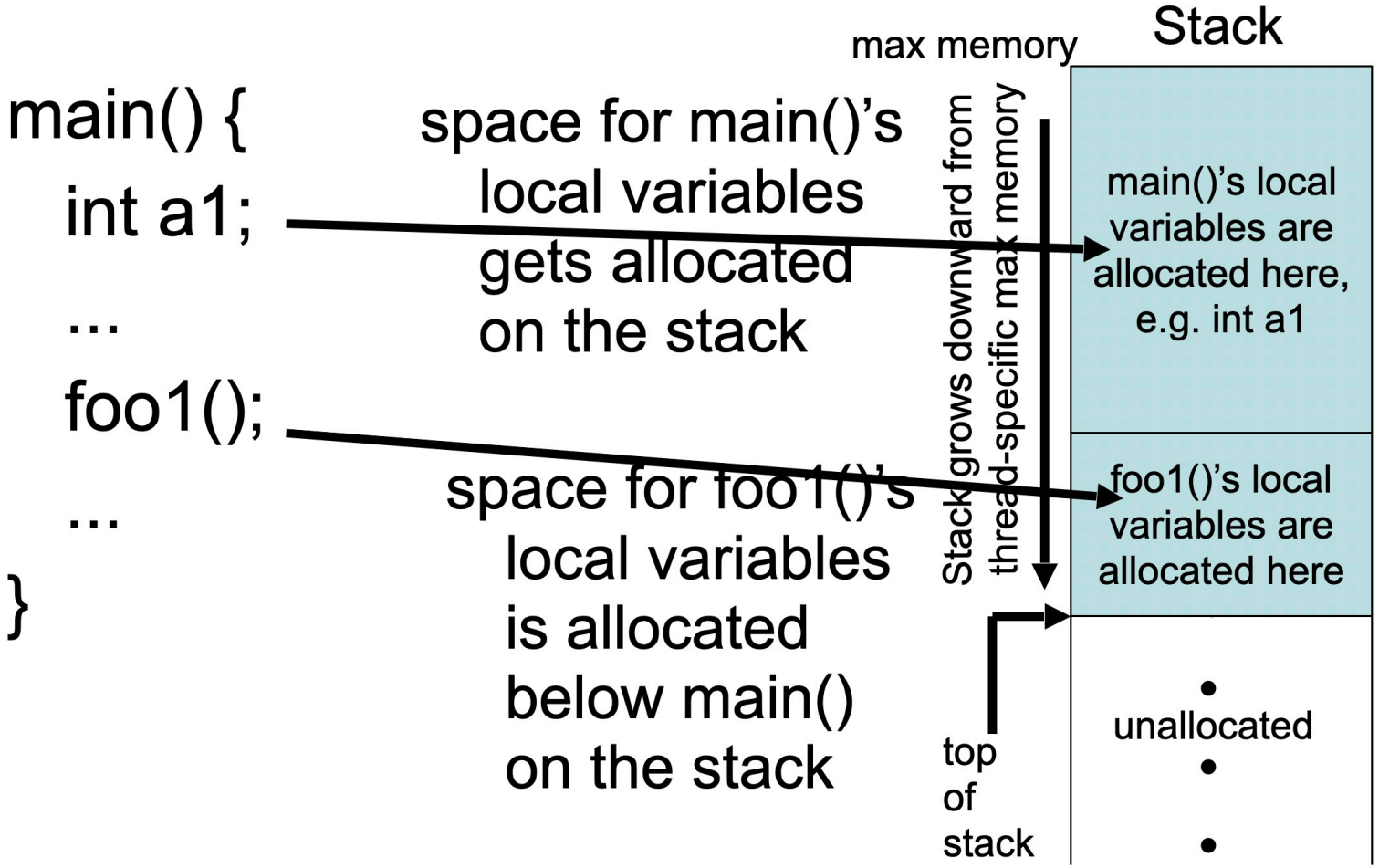


Computer Memory Utilization - Tools

- 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



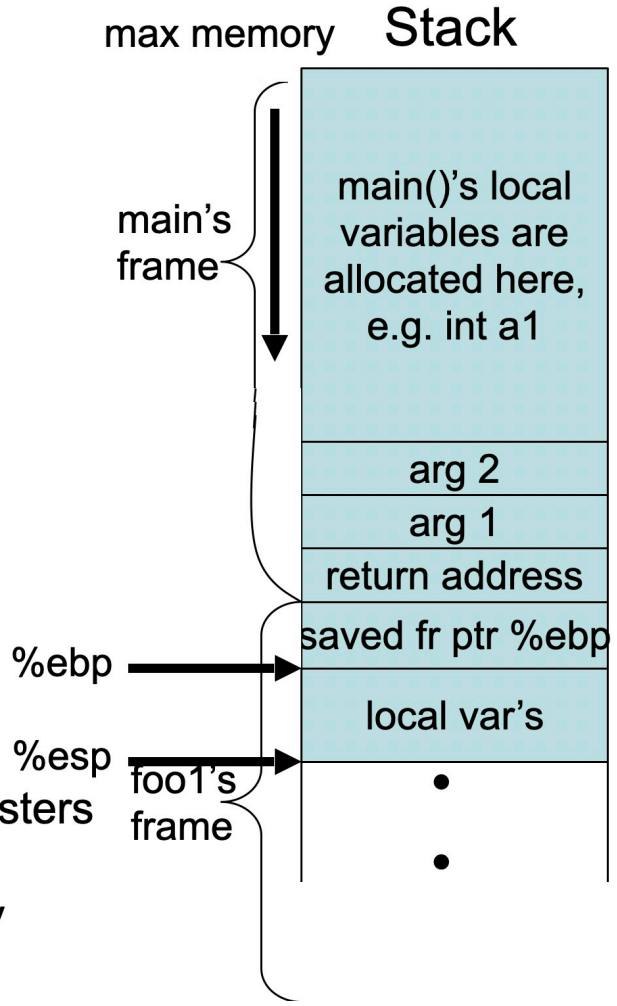
Source: CU Boulder - CSCI 3753 Operating Systems

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 83 - ??
  Chapter 4: Functions and Program Structure
  Chapter 5: Pointers and Arrays
  Chapter 6: Structures

- Lab Reading
  - Review Associates' Lab 1 Programs

Assignment - Lab Preparation

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	☆

Assignment - Lab Preparation

- Research Microchip Development Tools - Begin Install if Want

  MPLAB Harmony 3
  Embedded Software Development Framework for 32-bit Microcontrollers and Microprocessors

  https://www.microchip.com/en-us/tools-resources/configure/mplab-harmony

Look Ahead

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

Pomona College

Action Items and Discussion

AI#:	Owner	Slide #	Document	Action