Lab #2 Report: Memory Exploration

Nick Williams and Ethan Graney

10/06/2021

Screenshot + Components:

* Since the lab only involved Arduino IDE, the Arduino Uno Processor, and the code provided to us, there is no screenshot of the system.
* The only physical component in the lab was the AUP

Summary: The goal of this lab was to explore the Harvard architecture and its memory spaces, primarily focusing on the data memory. This was done by taking the code given to us by the TA and uncommenting the sections that correlate to the individual parts of the lab while an Ardunio was plugged into the computer. With each part of the lab, we were asked to analyze and record certain aspects of the memory to determine the type of system the Arduino is.

Results:

Part 1: In this section, we started the experiment by plugging the Ardunio into the computer and opening Arduino IDE for the code. We then copied the code from GitHub and placed it into the new sketch. After copying the code in its entirety, we when uncommented the “part 1” section of the code and ran it with the serial monitor visible. The three constraints found in the code were:

* End of Flash: 7fff
* End of RA: 8ff
* End of EEPROM: 3ff

Part 2: Since the code was already complete, we simply analyzed the arrays and determine how they were initialized. As with part 1 as with the latter two parts, we uncommented the “part 2” section of the code. After clearing the serial monitor and running the code, we were able to locate the arrays and pieced together how the compiler allocates the memory, which we did. Based on the output, the heap, stack, and initialized data ended up in 30E, 8F4, and 108 respectively.

Part 3: After seeing the locations of the arrays, we tested this out by printing out the sections of the RAM in this section. After uncommenting and running the “part 3” section of the code, it confirmed our hypothesis from part 2.

Part 4 (Optional): When re-commenting part 3 and uncommenting part 4, we were able to determine the type of machine the Arduino was, which was a Big Endian. We then were able to locate the long e and determine why the a and e in the Ram stack and not the heap. The location of the long e was at 8f5 because that is the location that it ended up. They are in the stack of the ram as the values of e and a represent dynamic memory allocation in the system. It’s only a temporary allocation instead of a globe allocation as they are defined in a method and are temporary.

Conclusions: In this lab, we learned how to determine whether a machine is a little or big endian, determine the constraints of a machine by running certain lines of code, and see where certain arrays end up in the data of a machine. All of this was possible due to the code provided to us in this lab. No mistakes were made in this lab as it was primarily code-based

Code:

// Part 1: print address ranges for each memory space

/\*

Serial.print("\nEnd of Flash: ");

Serial.println(FLASHEND, HEX);

Serial.print("End of RAM: ");

Serial.println(RAMEND, HEX);

Serial.print("End of EEPROM: ");

Serial.println(E2END, HEX);

Serial.println();

\*/

// Part 2: Arrays

/\*

Serial.println("String in program memory: ");

char c;

for (int i = 0; (c = pgm\_read\_byte(&array1[i])) != 0; i++) {

Serial.print(c);

}

Serial.println(array1); // accesses RAM - not program memory

Serial.print("Address of string in program memory: ");

Serial.println((int) &array1[0], HEX);

Serial.println();

Serial.println("String in RAM initialized data: ");

Serial.println(array2);

Serial.print("Address of string in RAM initialized data: ");

Serial.println((int) &array2[0], HEX);

Serial.println();

Serial.println("String in RAM stack: ");

Serial.println(array3);

Serial.print("Address of string in RAM stack: ");

Serial.println((int) &array3[0], HEX);

Serial.println();

Serial.println("String in RAM heap: ");

Serial.println(array4);

Serial.print("Address of string in RAM heap: ");

Serial.println((int) &array4[0], HEX);

Serial.println();

//Serial.println(array5);

Serial.println(array6);

\*/

// Part 3: print out the ram

/\* Example function calls:

displayRAM((char \*) 0x100, (char \*) 0x200, false);

//displays memory in 0x100 blocks with 2 second delays

displayAllRAM(2000, false);

\*/

//Part 4(OPTIONAL): What endian?

/\*

unsigned long a = 0x12345678;

unsigned long e = (unsigned long)&a;

Serial.print("Long location in RAM Stack :");

Serial.println(e, HEX);

displayRAM( (char \*)(e - 15), (char \*)(e + 15), true);

displayRAM( (char \*)(e - 15), (char \*)(e + 15), false);

\*/

}