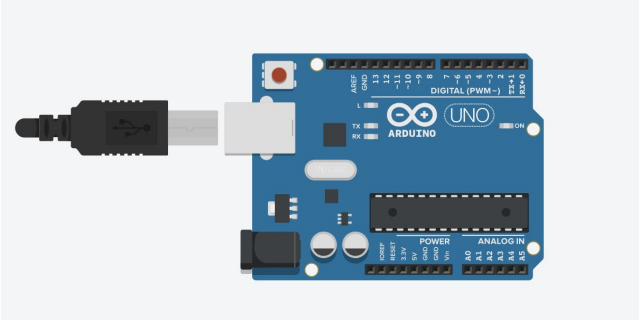
LAB REPORT 2

Names: Giancarlos Marte
Date: 2/21/21

Screenshot + components:



components

1. Bus: a set of wires that transfers information from one component to another.
2. Harvard Architecture: a computer design where different types of memory are stored in separate places.
3. Von Neumann Architecture: a computer design where different types of memory are stored in the same place.

Summary:

This lab was to learn more about memory and how it is stored. First I used the code in part 1 to figure out where the end of the Flash, RAM and EEPROM memory were. In part 2, I used the given code to see where different strings would be located in the RAM initialized data, RAM stack and RAM heap. Lastly in part 3 I had to figure out the memory addresses used by the initialized data, stack and heap. I used the functions given to me and did some trial and error to find the range of these memory types. The way I tried to figure out the range of them was by noticing where the string arrays data ended up based on the serial output.

Results:

Part 1:
End of Flash: 7FFF
End of RAM: 8FF
End of EEPROM: 3FF

Part 2:
String in program memory: Array 1
Address of string in program memory: 68 ← EEPROM
String in RAM initialized data: Array 2
Address of string in RAM initialized data: 106 ← EEPROM
String in RAM stack: Array 3

Address of string in RAM stack: 8F0 ← RAM
String in RAM heap: Array 4
Address of string in RAM heap: 33C ← EEPROM

Part 3:
Initialized data: starts at 0x100
RAM heap: starts at 0x110, 0x330 to 0x340 → starts at 0x110 and extends to 0x340
RAM stack: starts at 8F0

Conclusions:

1. I learned that in a Harvard architecture computer like the Arduino, different types of memory are located in different places.
2. I also learned how to access the addresses of certain data and where they were located in terms of memory type.

I did not have many mistakes as I did not have to write any code. However, I did have trouble with answering the question in part 3. I did not really understand what was considered the end for the initialized data, heap and stack. I assumed that wherever the array appeared was where it started. The only array that appeared in 2 places was heap, which I assumed was the start and end. The other two, I did not really know where they would end.

Code:

```
#include <avr/pgmspace.h>

//PROGMEM keyword creates the object in the flash memory instead of RAM
const char array1[] PROGMEM = "Array 1";

// put some strings in RAM initialized data area
char array2[] = "Array 2";
char array5[] = "Array 5"; //this will not be allocated memory if it is not used somewhere
char array6[] = "Array 6";


void setup() {
  Serial.begin(9600); // Setup for Serial output
  char array3[] = "Array 3"; // put a string in RAM stack area
  char *array4 = (char*) malloc(strlen("Array 4") + 1);
  strcpy(array4, "Array 4"); // put a string in RAM heap area


  // 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);
          */
  displayRAM((char *) 0x250, (char *) 0x350, false);
  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);
 */

}


void loop()  {} //notice we still have the loop function to keep the compiler happy

void displayAllRAM(int waitTime, bool hex) {
  for (char *i = (char *) 0x0; i < (char *) RAMEND; i += 0x100) {
        displayRAM(i, i + 0x100, hex);
        delay(waitTime);
        }
}

/* example call displayRAM((char *) 0x8E0, (char *) 0x8FF, true);
* if hex is false, characters will be printed, and all other values will be represented as '.'  */
void displayRAM(char *start, char *endd, bool hex) {
  char *array;
  for(array = start; array < endd; array += 0x10) {
        //create row number
        if (array < (char *)0x10)
        Serial.print('0');
        if (array < (char *)0x100)
        Serial.print('0');
        Serial.print((int)array, HEX);
        Serial.print(": ");
        //for each index (0 through 15 inclusive)
        for(int i = 0; i < 0x10; i++) {
        if(hex) {
        if (array[i] >= 0x00 && array[i] < 0x10)
        Serial.print('0');
        Serial.print(array[i] & 0xFF, HEX); //0xFF is our bitmask
        } else {
        Serial.print((array[i] >= ' ' && array[i] <= 'z') ? array[i] : '.');
        }
        Serial.write(' ');
        }
        Serial.println();
  }
}
```

Rubric:

Each lab is graded out of 10. Labs are due at midnight a week after they are assigned. Labs turned in late receive a max of 7 points:

Item	Points worth
Code correctness	3
Submission form correct	3
Report contains accurate information	2
Some effort put into report*	2

*No answer is too short to properly address the lab report section and I can tell you tried at least just a little.