Faculty of Science, Engineering and Technology



Computer Systems

Week 9

***PART 9.1: Indirect and Indexed Addressing***

A screenshot of a computer

Description automatically generated**Question 9.1.1.a: Write a simple ARMlite assembly program that draws a single line of the same length across the second row (starting from the left-most column) in Low-res display mode**

**Question 9.1.1.b: Add to your assembly program code that draws a single line of the same length vertically, down the middle of the display in Low-res display mode**

A blue screen with white text

Description automatically generated

A screenshot of a computer

Description automatically generated **Question 9.1.2:**

**Question 9.1.3.a: Explain what specifically makes this code an example of indirect addressing? How is it using indrect addressing to draw each pixel?**

1. Line 1 “MOV R1, #.PixelScreen”: Moves the address of the pixel display memory into register R1.
2. Line 2 “MOV R2,#.red”: Stores the value representing red into register R2. The label “#.red” probably corresponds to a predefined value for the red color (such as a specific hexadecimal value).
3. Line 3 “MOV R3,#0”: Indicates that store the value 0 to register R3
4. Line 4 “loop:”: Marking the beginning of the loop
5. Line 5 “ADD R4,R1,R3”: This calculates the effective address for the next pixel by adding the base address (R1) and the offset (R3) which the result is stored in R4
6. Line 6 “STR R2,[R4]”: Stores the color from R2 (red) into the address pointed to by R4.
7. Line 6 “ADD R3,R3,R4”: Increments the offset R3 by 4. This accounts for the width of a pixel, or the amount of memory allocated per pixel (4 bytes, possibly for an RGBA format).
8. Line 8 “CMP R3,#80”: Compares the offset R3 to 80, which likely represents a width of 80 pixels.
9. Line 9 “BLT loop”:  If R3 is less than 80, control jumps back to the start of the loop, repeating the pixel coloring process.
10. Line 10 “HALT”: Means end the program

A computer screen shot of a blue screen

Description automatically generated **Question 9.1.3.b:**

A screenshot of a computer

Description automatically generated **Question 9.1.3.c:**

***PART 9.2: Indexed Addressing***

A computer screen shot of a blue and white computer

Description automatically generated**Question 9.2.1:**

A computer screen shot of a blue and white computer

Description automatically generated **Question 9.2.2:**

***PART 9.3: Arrays***

**Question 9.3.1.a:**

**-** The purpose of the “.Align 256” instruction aligns the next data or code segment to a 256-byte boundary in memory. This improves access speed and ensures capabilities with certain architectures that require aligned data for optimal performance. It adds padding bytes as needed to achieve this alignment.

- The error on Line 2 (arrayLength: 10) occurs because the assembler interprets “arrayLength: 10” as an introduction rather than a data declaration:

+ Memory State: After the “. ALIGN 256” directive, the assembler expects a valid instruction or data declaration to follow. If arrayLength is not defined properly (e.g., as a data directive), the assembler will treat 10 as an instruction, leading to an "Unknown Instruction" error.

+ Missing Data Directive: To properly declare arrayLength, you need to use a directive (like .data or similar) to inform the assembler that you're defining data rather than executable code.

**Question 9.3.1.b:**

A computer screen shot of a blue and white computer

Description automatically generated

A blue screen with white text

Description automatically generated **Question 9.3.1.c:**

A blue screen with white text

Description automatically generated**Question 9.3.2:**

***PART 9.4: More Arrays***

A screenshot of a computer

Description automatically generated**Question 9.4.1**

A screenshot of a computer

Description automatically generated**Question 9.4.2:**