Part 1:

Item 1:

Address: 0x1001001c

Item 2:

lui $1,0x00001001

ori $8,$1,0x00000018

Item 3:

0x3c011001

0x34280018

The last 4 bits are the same.

Part 2:

2.1 Strings and Data

String constant: MINIMUM: .asciiz "minimum: " (line 22)

First four HEX: 6D696E69

Address: 0x10010043

Part 3:

Something new I learned from this exercise is the thought process of searching for accurate processes in the program instructions by looking for data. Assembly represents data such as strings using hexadecimal when appearing in an address registry. With MARS allowing the steps of a program to be reversed incrementally, it was helpful to see what is happening in memory. This will be useful in the future during the debugging process to confirm that specific values are prepositioned and ready for the instructions to function as intended.

To find each line for this exercise, I’d start by scrolling through the Text Segment of the assembled program and confirm the number in the source column matched the source file. I would step through the instructions and follow any indicated changes in the register. Over on the tab displaying the resisters, I would notice matching values between the hex encoding and the addresses in the data segment. Specifically, when I searched for the string constant, I looked up the hexadecimal encoding and found it located at the exact address referenced as a register value in the previous step.

A screenshot of a computer

Description automatically generated

One question inspired by this exercise is the function of the last 3 registers {pc, hi, lo}. These registers don’t seem accessible with MIPS so it seems like a redundant allocation of processing space.