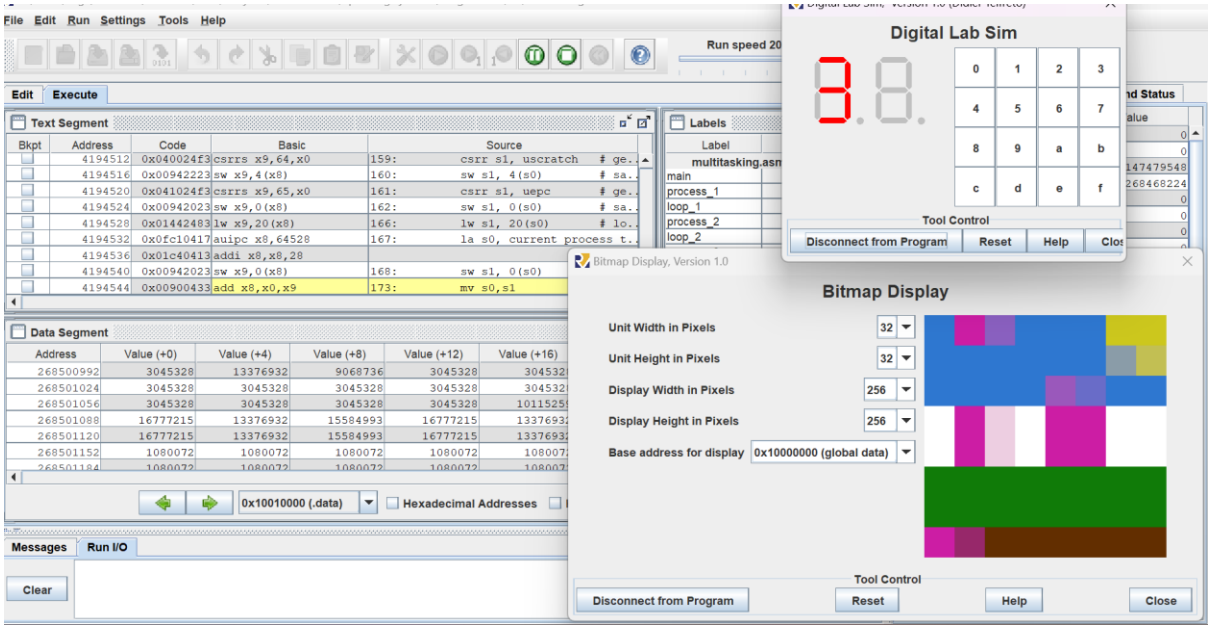
TeamTontos Assignment 3 Joint Report. Fionn Murphy (21349726) and Sergio Jimenez (21710801)

Q1)

For Q1, Multitasking, we started off with the code in the file multitasking.asm and opened the Bitmap Display and the Digital Lab Sim Tool. Then we went over to GIMP and painted a beautiful picture (8x8 image) and exported it as raw data. After that, we converted the image data into the list of 64 hexadecimal pixel colors in Ubuntu using the command: hexdump -v -e '/3 ".word 0x"' -e '3/1 "%02x"' -e '/3 "\n"' man.data.

Code wise, we used .data to put the data into an array-like structure. Then we created trap_frame_3 to "connect" the newly created process_3 to the already existing loop. In process_3, we loaded the address of the data into s0, and loaded the addresses of the Bitmap Display (0x10000000) in s1 and the address of the filled Bitmap Display (0x10000100) in s3. Then we created the label "loop_3" and "end". In the loop, we load the first word into S2 and then put it from S2 into the Bitmap Display. After that, we add 4 to the address of the Bitmap Display and 4 to the address of the array of words, so that in the next loop iteration the next word is loaded into the next position in the Bitmap Display. After adding 4 into s1 and s0, we check if the new s1 is greater than s3 (the address of the filled Bitmap Display) and if it is, the program goes to the label "end" and finishes process_3. If not, it loops back to loop_3.



Q2)

For question 2 we start by creating a 2d array of ints that will act as our three pipes. The byte we want to pass is the char ‘a’.

We create two child processes using fork() and in each of these processes we loop a given number of times, reading from the pipe output it has with the process before it and writing to the input of the

pipe it shares with the process after it in the cycle. Once it has completed all its loops it exits so that the process can stop.

We start the looping by passing our byte into the input our parent shares with a process. This causes that process to read the output of its pipe and this process repeats until the loop has completed 10,000 times.

To time how many loops our code completes in a second we used a stopwatch and found that it took 7.45 seconds to complete 10,000 loops which equates to 1,342 loops per second.

```
$ assignment3q2
Done
$
```

Q3)

For this question we first use ‘cut’ to return delimited (-d) elements separated by spaces (‘ ‘) on each line. -f1 selects only the first element and –complement returns every element that’s not selected from sample.txt. We then give this output to tr which we use to replace every space with a newline character to format the output on new lines. We use -s so that if there are multiple spaces between inputs we only replace them with one newline character.

This is our command:

cut -d ' ' -f1 --complement sample.txt | tr -s ' ' '\n'

```
fionn@LAPTOP-R1KQ9435:~$ cut -d ' ' -f1 --complement sample.txt | tr -s ' ' '\n'
54
68
69
73
20
69
73
20
61
6e
20
65
78
61
6d
70
6c
65
20
6f
66
20
68
65
78
20
64
75
6d
70
45
11
98
20
20
20
20
20
20
fe
ed
ba
ad
0d
0a
fionn@LAPTOP-R1KQ9435:~$
```