SCC150 – MIPS/Assembly Week 12 Practical

Angie Chandler

SCC – Lancaster University

Week 12

Outline

- Bitmap Display
- Using lui and sw
- Displaying memory in colour
- Creating a counter
- Drawing a picture!

Why Draw Pictures in MIPS?

- Two reasons
  - It's a good way to practice memory access – you can see what you've done!
  - The original PlayStation actually used MIPS, so in honour of that…
    - PS1 is quite old, so to give you an idea, my favourite PS1 game was Wipeout 2097, see image and video for the curious
    - https://www.youtube.com/watch?v=GmTRFf4Pquk

Bitmap Display Introduction

- Open MARS
- In MARS, go to Tools on the Menu and select Bitmap Display
- Press the button that says Connect to MIPS on Bitmap Display
- You are now connected to the display…

- Notice that it says “base address for display” is 0x10010000
- This means that anything stored in that register will be represented as colour on the display

Accessing Memory for Display (lui and sw)

```
#address of memory for image (lui – load upper immediate)
addi $s0,$zero,0          #sets $s0 to zero – initialise
lui $s0, 0x1001           #loads upper register with 0x1001
#this sets register $s0 to 0x10010000 (address of memory for bitmap)

#set the content of the register at address 0x10010000
addi $a0, $zero,0x00ff    #set register $a0 to 0xff
sw $a0, 0($s0)            #store in $s0
```

Colour!

- Congratulations! You've now coloured a pixel in blue!
  - Of course, it's one pixel, so you can't see it…
  - But the register shows it

- How do I know it's blue?
  - RGB – look up hex colours online
  - 0x00000000
  - You need to either use lui or sll to reach the red bits

Exercise 1 – Visible Colour

- Obviously, you can't see just one pixel, so you need to keep going.
- Remember word size, each time you move to the next register, you need to add 4
- Note, what you want to do is:

```
sw $a0, 4($s0)          #store contents of $a0 in $s0+4
```

- But you can't (the number (i.e. 4) can't be a variable), so you have to add 4 to the base address

Exercise 2 – Background Colour

- Fill in the entire background in a colour of your choosing
- Is it stopping in the right place, or are you going too far?
- 512 x 256 pixels, 4 bytes per word
- Counter
- Branch - to detect the end

- Got it working? Try a different colour!

Exercise 3 – Draw a Line

- This requires better control
- Fill in the background, then draw a line in a different colour.
- Start with a horizontal line
- Then try a vertical line.

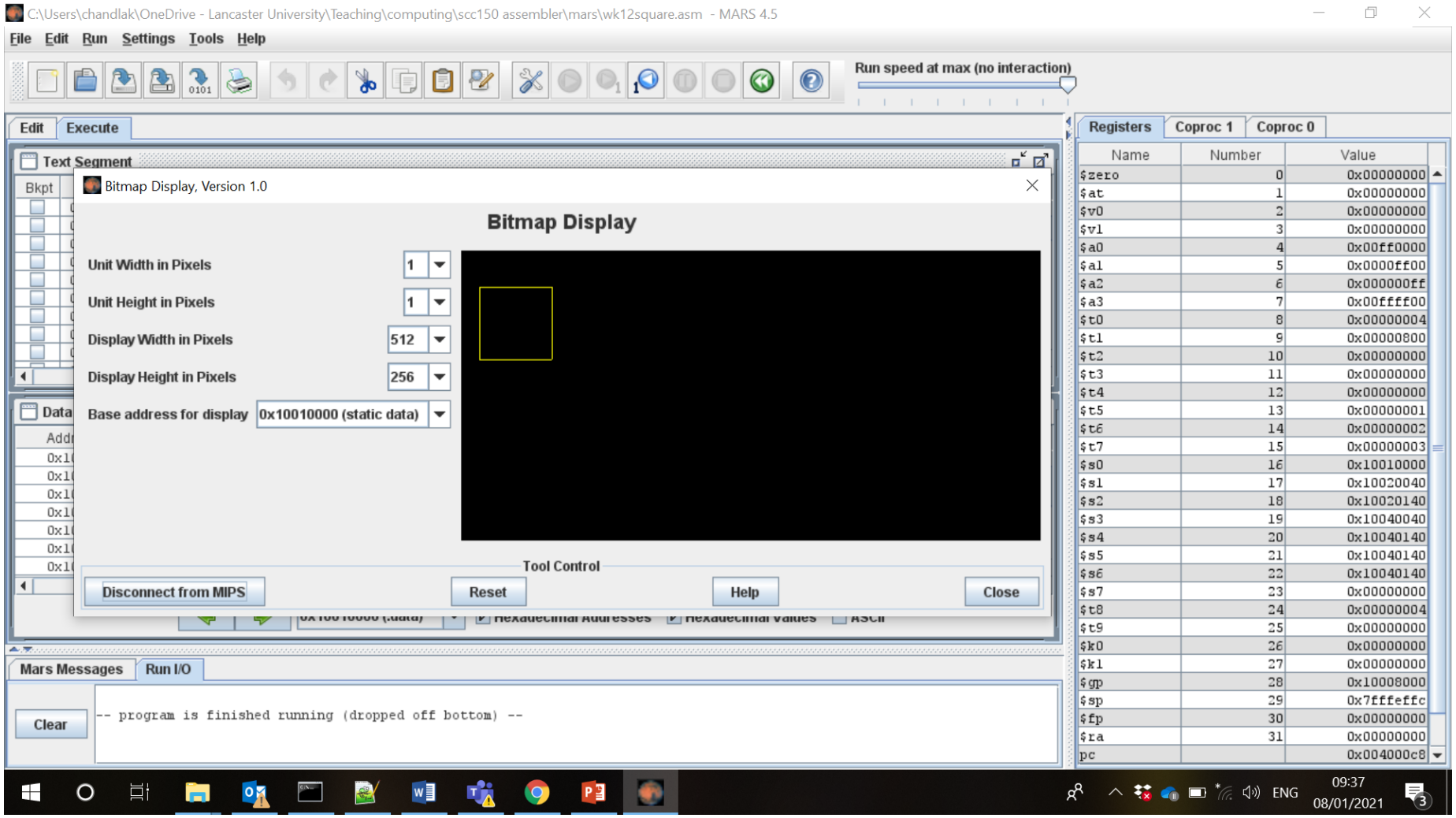
- Remember 512 x 256 pixels, 4 bytes per register

Exercise 4 (optional) – Draw a Square

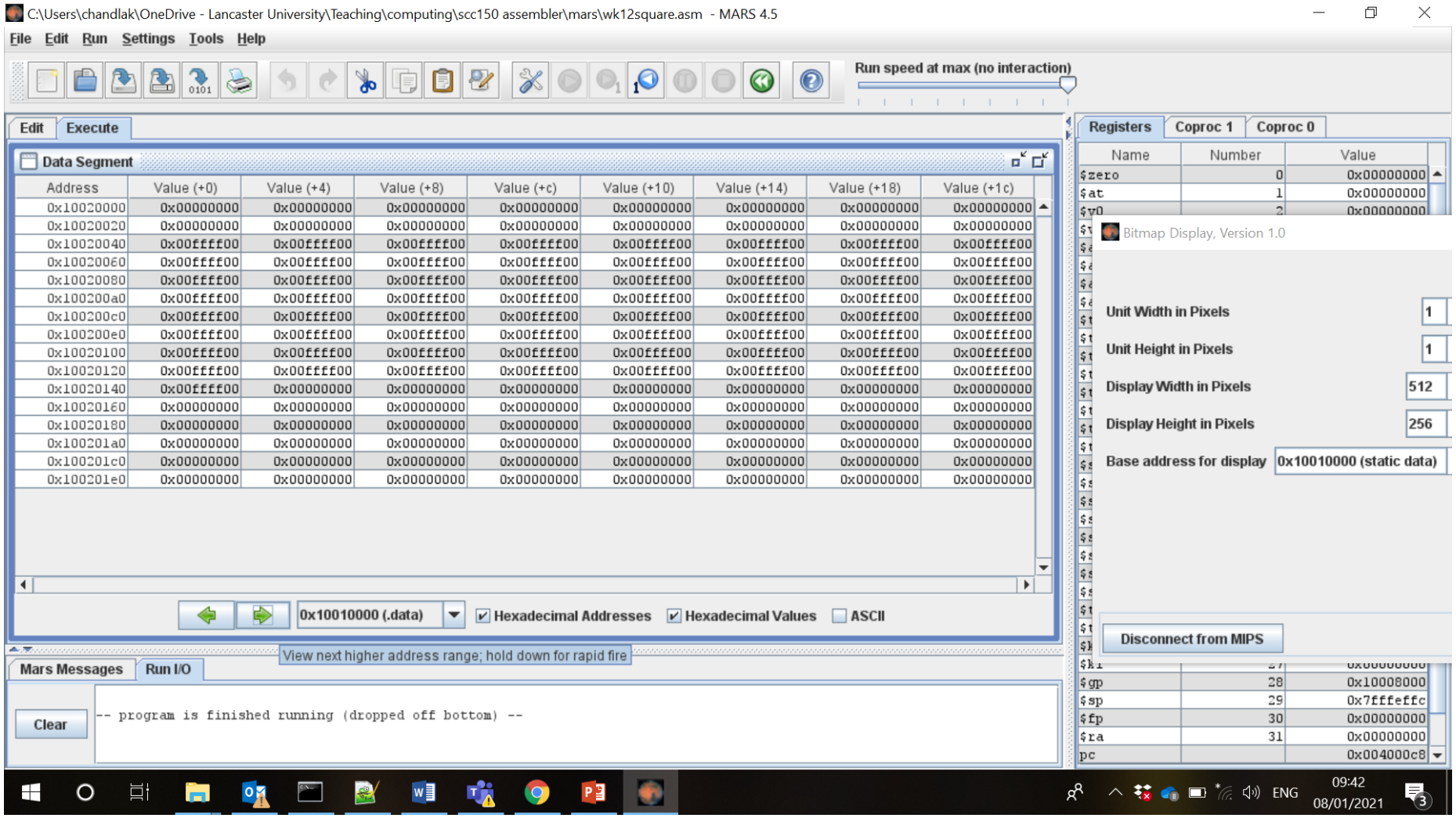
- Extend what you've learnt to draw a square on the display
- How might you draw multiple squares?

- How might you set the program up to change colours easily?

Examples – Drawing



Examples – Drawing



Note – the green arrows allow you to move along the memory

See the non-zero memory addresses? Essentially, that's all you're trying to do