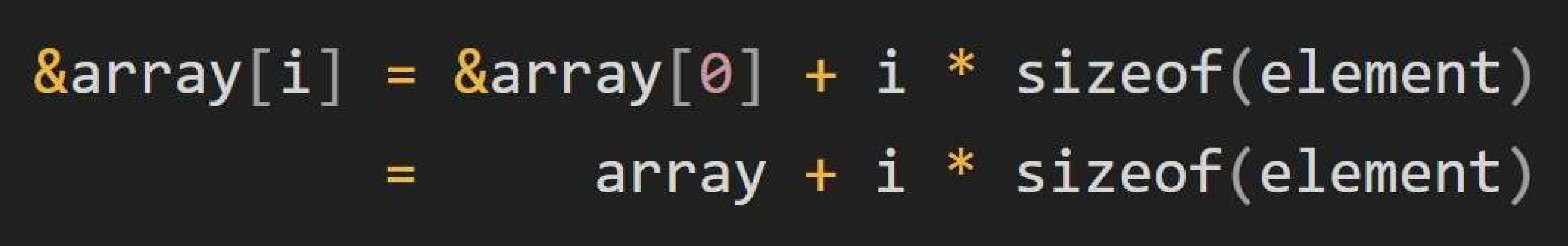
Week 4!

2D Arrays & Functions

1D Arrays

- Last week we saw that the formula for calculating the memory address for 1D arrays was:

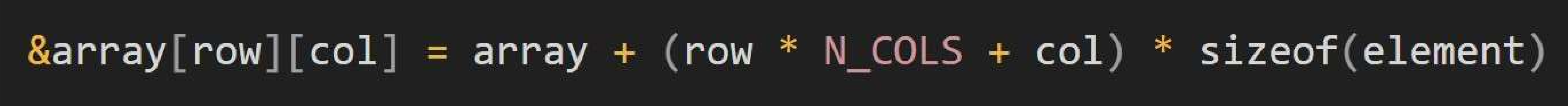


```
&array[i] = &array[0] + i * sizeof(element)
          =     array + i * sizeof(element)
```

2D Arrays

- What about 2D arrays? What do they look like in memory?
    - 2D Array Spreadsheet

- For 2D arrays, the formula is:



```
&array[row][col] = array + (row * N_COLS + col) * sizeof(element)
```

- Example: Tutorial Q2

Functions

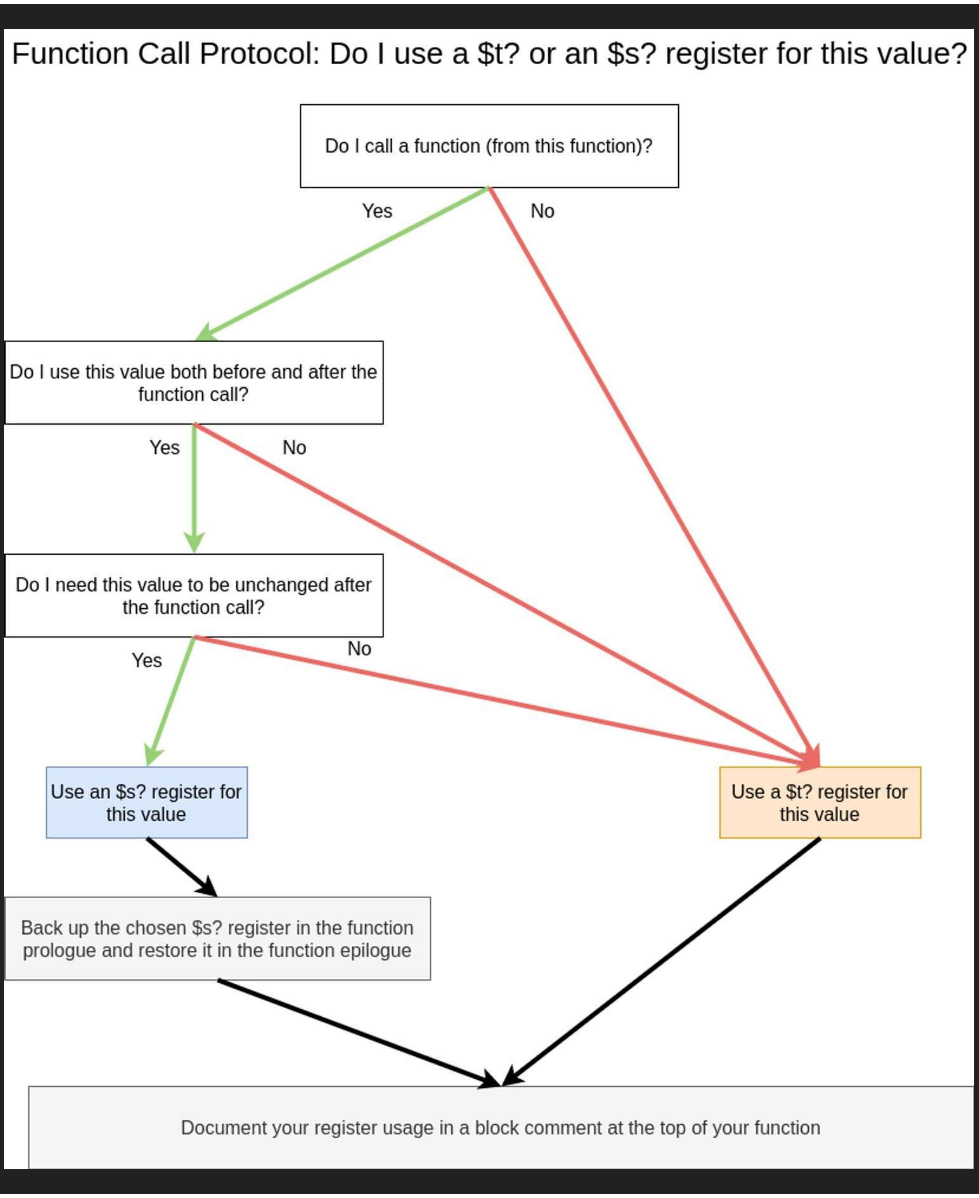
- Functions are really just fancy labels which we can jump to and return from
- We use a special instruction called **jal**
    - This is just like **j** but it updates the value of **$ra** to point to the next instruction after the jump
- But this presents us with an issue - we get infinite loops
- We solve this using the **stack**
    - **push**, **pop**, **begin**, **end**

Function Calling Conventions

- **Treat functions as black boxes!** (pretend you don’t know how it works)

- **$a**: arguments, may be overwritten
- **$v0**: return value upon completion
- **$t**: local variables, may be overwritten (assume ALL are destroyed)
- **$s**: local variables, assume that they are NOT overwritten
    - We don’t get this for free! If our function modifies any **$s** registers, we have to restore them to their original values before our function returns

$s or $t?



Function Call Protocol: Do I use a $t? or an $s? register for this value?

Do I call a function (from this function)?

Yes

No

Do I use this value both before and after the function call?

Yes

No

Do I need this value to be unchanged after the function call?

Yes

No

Use an $s? register for this value

Use a $t? register for this value

Back up the chosen $s? register in the function prologue and restore it in the function epilogue

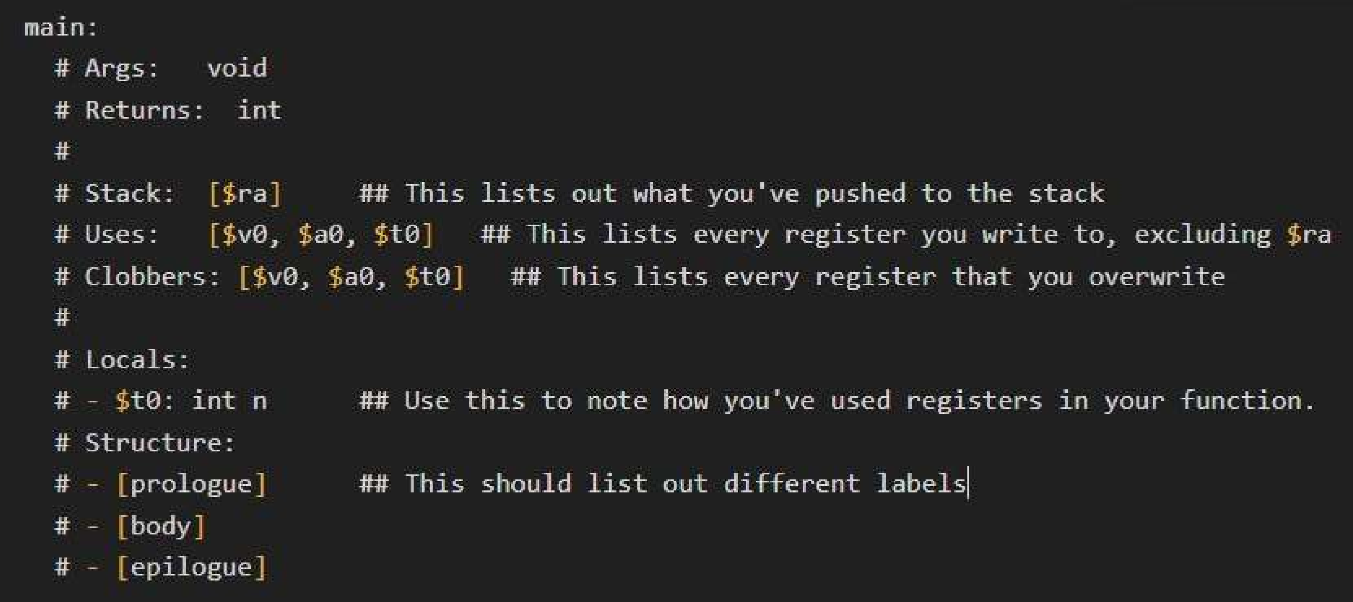
Document your register usage in a block comment at the top of your function

Assignment 1

- Comments

  (C & function)
- Labels

- Do easiest functions first
- Keep track of registers



```
main:
  # Args:   void
  # Returns:  int
  #
  # Stack:  [$ra]     ## This lists out what you've pushed to the stack
  # Uses:   [$v0, $a0, $t0]   ## This lists every register you write to, excluding $ra
  # Clobbers: [$v0, $a0, $t0]   ## This lists every register that you overwrite
  #
  # Locals:
  # - $t0: int n      ## Use this to note how you've used registers in your function.
  # Structure:
  # - [prologue]      ## This should list out different labels
  # - [body]
  # - [epilogue]
```