

# Boolean Algebra + VM

CIS 2400 Recitation 1

# Recitation Outline

- Boolean Algebra
  - Truth Tables
  - Boolean Rules
  - Simplification
- Bitwise Operations
- Basic VM Tutorial and Setup (Demo)

# Boolean Algebra

# Truth Tables

| $P$ | $\neg P$ |
|-----|----------|
| T   | F        |
| F   | T        |

| $P$ | $Q$ | $P \wedge Q$ |
|-----|-----|--------------|
| T   | T   | T            |
| T   | F   | F            |
| F   | T   | F            |
| F   | F   | F            |

| $P$ | $Q$ | $P \vee Q$ |
|-----|-----|------------|
| T   | T   | T          |
| T   | F   | T          |
| F   | T   | T          |
| F   | F   | F          |

# Exercises

| P | Q | R | Q & R | $P \parallel (Q \& R)$ |
|---|---|---|-------|------------------------|
|   |   |   |       |                        |
|   |   |   |       |                        |
|   |   |   |       |                        |
|   |   |   |       |                        |
|   |   |   |       |                        |
|   |   |   |       |                        |
|   |   |   |       |                        |
|   |   |   |       |                        |
|   |   |   |       |                        |

# Exercises

| P | Q | R | Q & R | $P \parallel (Q \& R)$ |
|---|---|---|-------|------------------------|
| t | t | t | t     | t                      |
| t | t | f | f     | t                      |
| t | f | t | f     | t                      |
| t | f | f | f     | t                      |
| f | t | t | t     | t                      |
| f | t | f | f     | f                      |
| f | f | t | f     | f                      |
| f | f | f | f     | f                      |

# Exercises

| P | Q | R | $\sim R$ | $Q \ \& \ \sim R$ | $P \ \& \ (Q \ \& \ \sim R)$ |
|---|---|---|----------|-------------------|------------------------------|
|   |   |   |          |                   |                              |
|   |   |   |          |                   |                              |
|   |   |   |          |                   |                              |
|   |   |   |          |                   |                              |
|   |   |   |          |                   |                              |
|   |   |   |          |                   |                              |
|   |   |   |          |                   |                              |
|   |   |   |          |                   |                              |
|   |   |   |          |                   |                              |

# Exercises

| P | Q | R | $\sim R$ | $Q \& \sim R$ | $P \& (Q \& \sim R)$ |
|---|---|---|----------|---------------|----------------------|
| t | t | t | f        | f             | f                    |
| t | t | f | t        | t             | t                    |
| t | f | t | f        | f             | f                    |
| t | f | f | t        | f             | f                    |
| f | t | t | f        | f             | f                    |
| f | t | f | t        | t             | f                    |
| f | f | t | f        | f             | f                    |
| f | f | f | t        | f             | f                    |



## Quick Note on Notation

We will use C bitwise notation in these slides to be consistent with lecture!

- $\&$  = AND
- $|$  = OR
- $\sim$  = NOT

May be more intuitive to think with mathematical notation (we'll see why later)

- $*$  = AND
- $+$  = OR
- $\bar{A}$  = NOT

# Boolean Rules

## ❖ Identity

- $A \& 1 = A$
- $A \& 0 = 0$
- $A \mid 1 = 1$
- $A \mid 0 = A$
- $\sim\sim A = \text{NOT NOT } A = A$

## ❖ Associative

- $A \& (B \& C) = (A \& B) \& C$
- $A \mid (B \mid C) = (A \mid B) \mid C$

## ❖ Distributive

- $A \& (B \mid C) = (A \& B) \mid (A \& C)$
- $A \mid (B \& C) = (A \mid B) \& (A \mid C)$

## ❖ More Identity

- $A \& A = A$
- $A \mid A = A$
- $A \& \sim A = 0$
- $A \mid \sim A = 1$

*More on De Morgan's later*

## ❖ De Morgan's Law

- $\sim(A \& B) = \sim A \mid \sim B$
- $\sim(A \mid B) = \sim A \& \sim B$

# Boolean Simplification

Using the rules from the previous slide,  
we can simplify Boolean patterns

$$\text{e.g. } (A \& B) \mid (A \& \sim B)$$

$$= A \& (B \mid \sim B)$$

$$= A \& 1$$

$$= A$$

- |  |   |
|--|---|
| <ul style="list-style-type: none"> <li>❖ Identity           <ul style="list-style-type: none"> <li>▪ <math>A \&amp; 1 = A</math></li> <li>▪ <math>A \&amp; 0 = 0</math></li> <li>▪ <math>A \mid 1 = 1</math></li> <li>▪ <math>A \mid 0 = A</math></li> <li>▪ <math>\sim\sim A = \text{NOT NOT } A = A</math></li> </ul> </li> <li>❖ Associative           <ul style="list-style-type: none"> <li>▪ <math>A \&amp; (B \&amp; C) = (A \&amp; B) \&amp; C</math></li> <li>▪ <math>A \mid (B \mid C) = (A \mid B) \mid C</math></li> </ul> </li> <li>❖ Distributive           <ul style="list-style-type: none"> <li>▪ <math>A \&amp; (B \mid C) = (A \&amp; B) \mid (A \&amp; C)</math></li> <li>▪ <math>A \mid (B \&amp; C) = (A \mid B) \&amp; (A \mid C)</math></li> </ul> </li> </ul> | <ul style="list-style-type: none"> <li>❖ More Identity           <ul style="list-style-type: none"> <li>▪ <math>A \&amp; A = A</math></li> <li>▪ <math>A \mid A = A</math></li> <li>▪ <math>A \&amp; \sim A = 0</math></li> <li>▪ <math>A \mid \sim A = 1</math></li> </ul> </li> <li style="text-align: center; color: #c00000; font-size: small; margin: 5px 0;"><i>More on De Morgan's later</i></li> <li>❖ De Morgan's Law           <ul style="list-style-type: none"> <li>▪ <math>\sim(A \&amp; B) = \sim A \mid \sim B</math></li> <li>▪ <math>\sim(A \mid B) = \sim A \&amp; \sim B</math></li> </ul> </li> </ul> |
|--|---|

# Exercises

Simplify the following

$$1. \quad \sim(\sim A \mid \sim B)$$

$$A \ \& \ B$$

$$2. \quad \sim A \ \& \ B \ \& \ \sim C \ \mid \ \sim A \ \& \ B \ \& \ C \ \mid \ A \ \& \ \sim B \ \& \ C \ \mid \ A \ \& \ B \ \& \ C$$

$$\sim A \ \& \ B \ \mid \ \sim A \ \& \ C$$

$$3. \quad \sim(\sim A \ \& \ B \ \mid \ A \ \& \ C)$$

$$(A \ \mid \ \sim B) \ \& \ (\sim A \ \mid \ \sim C)$$

## ❖ Identity

- $A \ \& \ 1 = A$
- $A \ \& \ 0 = 0$
- $A \ \mid \ 1 = 1$
- $A \ \mid \ 0 = A$
- $\sim\sim A = \text{NOT NOT } A = A$

## ❖ Associative

- $A \ \& \ (B \ \& \ C) = (A \ \& \ B) \ \& \ C$
- $A \ \mid \ (B \ \mid \ C) = (A \ \mid \ B) \ \mid \ C$

## ❖ Distributive

- $A \ \& \ (B \ \mid \ C) = (A \ \& \ B) \ \mid \ (A \ \& \ C)$
- $A \ \mid \ (B \ \& \ C) = (A \ \mid \ B) \ \& \ (A \ \mid \ C)$

## ❖ More Identity

- $A \ \& \ A = A$
- $A \ \mid \ A = A$
- $A \ \& \ \sim A = 0$
- $A \ \mid \ \sim A = 1$

*More on De Morgan's later*

## ❖ De Morgan's Law

- $\sim(A \ \& \ B) = \sim A \ \mid \ \sim B$
- $\sim(A \ \mid \ B) = \sim A \ \& \ \sim B$

# Bitwise Operations

# Bitwise Operations Overview

- Various operations can be performed on bits in C
  - **&**
    - Bitwise AND
      - $0x9 \ \& \ 0x3 = 0x1$
      - $0b1001 \ | \ 0b0011 = 0b0001$
  - **|**
    - Bitwise OR
      - $0xA \ | \ 0x9 = 0xB$
      - $0b1010 \ | \ 0b1001 = 0b1011$
  - **^**
    - Bitwise XOR
      - $0x3 \ ^ \ 0xD = 0xE$
      - $0b0011 \ ^ \ 0b1101 = 0b1110$

# Bitwise Operations Overview

- ❖ Various operations can be performed on bits
  - $\sim$ 
    - Bitwise NOT or “complement”
      - $\sim 0x5 = 0xA$
      - $\sim 0b0101 = 0b1010$
  - $\ll$ 
    - Logical Left shift
      - $0x2 \ll 2 = 0x8$
      - $0b0010 \ll 2 = 0b1000$
  - $\gg$ 
    - Right shift (arithmetic if signed, logical if unsigned)
      - $0x4 \gg 1 = 0x2$
      - $0b0100 \gg 1 = 0b0010$

# Bit-Puzzles!

1. Write a function *isEven(int x)* that returns 0 if *x* is even, and 1 if *x* is odd

```
int isEven(int x) {  
    int result;  
    if (x & 1 == 1) {  
        result = 1;  
    } else {  
        result = 0;  
    }  
    return result;  
}
```



# No if statements? No problem

```
int isEven(int x) {  
    return x & 1;  
}
```

## Bit-Puzzles!

2. Write a function *isNegative(int x)* that returns 1 if *x* is negative, and 0 if *x* is positive (Note: int size is 4 bytes = 32 bits)

```
int isNegative(int x) {  
    return (x >> 31) & 1;  
}
```

## Bit-Puzzles!

3. Write a function `clearBit(int x, int n)` that clears the *n*th bit of the integer `x` (clear = set to 0)

```
int clearBit(int x, int n) {  
    return x & ~(1 << n);  
}
```

## Bit-Puzzles!

4. Write a function `toggleBit(int x, int n)` that toggles the *n*th bit of the integer *x* (toggle = flip the bit)

```
int toggleBit(int x, int n) {  
    return x ^ (1 << n);  
}
```

# VM Demo + Command Line Basics

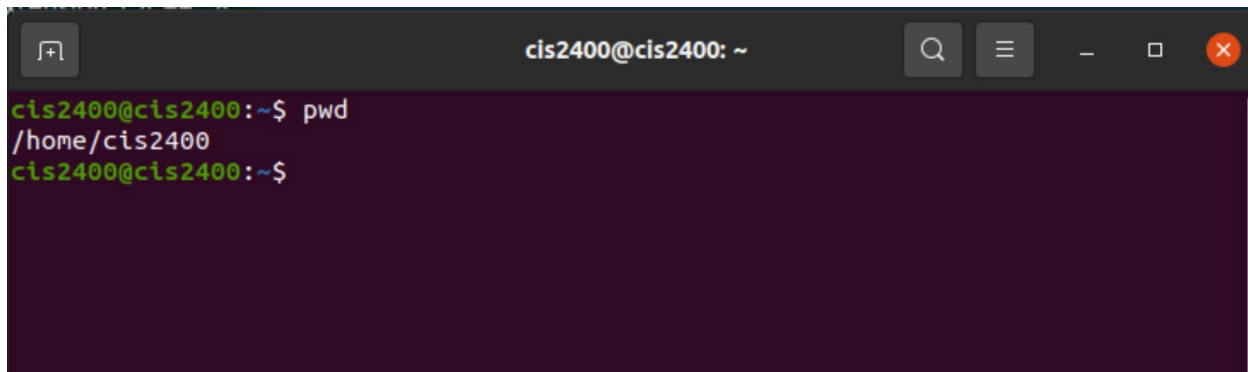
(slides adapted from [Prof. Harry Smith's CIS19x slides](#))

## Agenda for the VM Demo

- Unix command line basics
- Quality of life enhancements for the VM (optional)
  - Git basics
  - Installing VS Code
  - Increasing memory on the VM
- Tips on compiling HW1

# Unix Command Line

- A simple, text-based interface to the computer



```
cis2400@cis2400: ~  
cis2400@cis2400:~$ pwd  
/home/cis2400  
cis2400@cis2400:~$
```

- One of many ways to interact with a computer; enduring because of how easy it is to work with text.

## Basic Interaction

- Text appears at the bottom of the screen
- up/down arrows scroll through command history
- tab autocompletes when possible
  - if `g` could complete to `ga la` or `granny-smith`, then pressing tab twice would show both options



# Commands, pt 1

- `clear` - clear the screen
- `ls` - list contents of the current directory
- `pwd` - print working directory

# Commands, pt 2

- `cd` - change directory
  - `cd ..` - go up one directory
  - `cd dir_name` - go into directory called `dir_name`

# Quality of life improvements for the VM (optional)

- Problem: Shared folder doesn't work on the VM for M1 / M2 Macs :(ul>  - Solution: Set up a private github repo for each HW!
- Problem: I don't like writing code in Sublime Text!
  - Solution: Install VS Code & the C / Makefile plugins on the VM!
- Problem: VM is slow!
  - Solution: Increase the memory of the VM!
- Note: all of these solutions are completely optional!
  - You can still use the CIS 2400 VM without doing any of this

# Why use Git?

- Version control
  - You can keep track of what changes you made at a given time
  - You can undo changes you made
- You can transfer files between the VM and your local laptop if the shared folder doesn't work on the VM

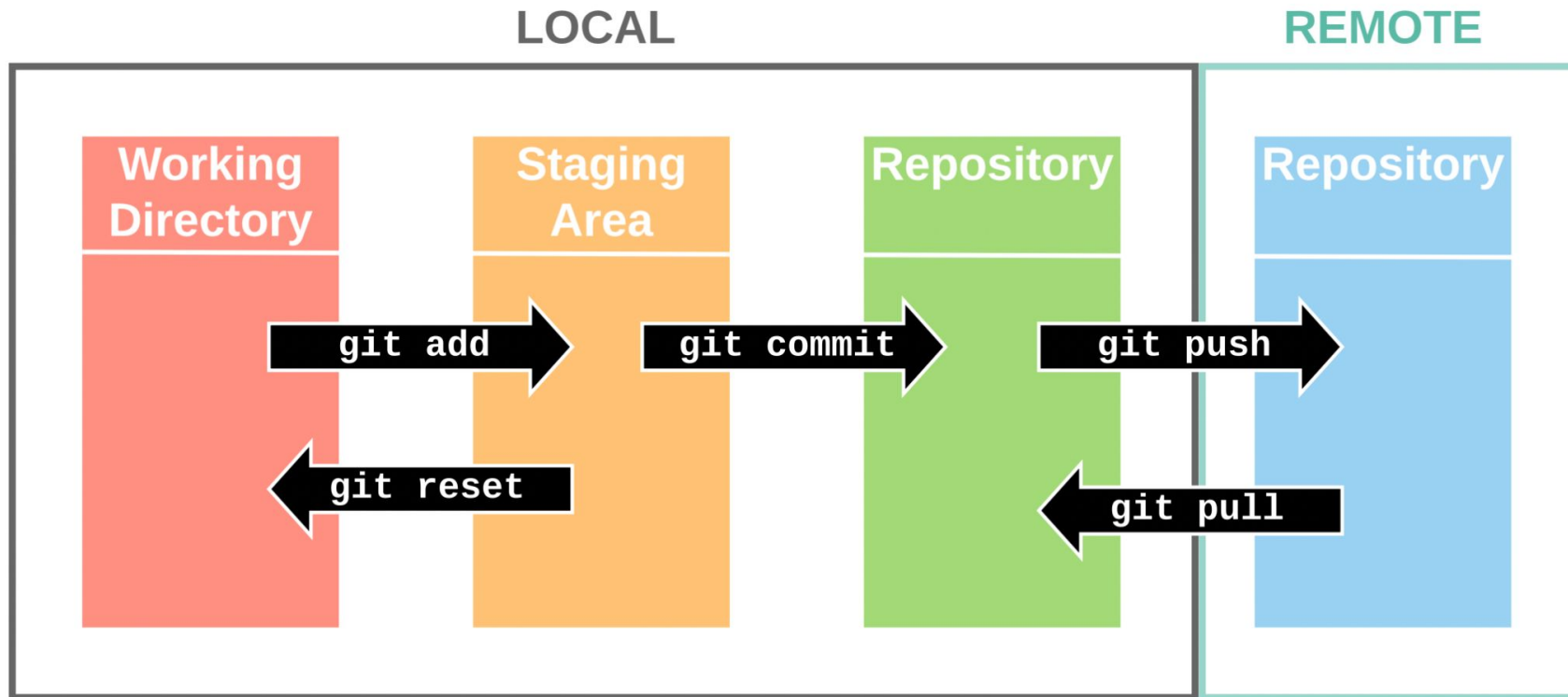
## Bringing in GitHub

- Can create a repo on GitHub, but at first it has nothing to do with the local one on your machine!
- Need to `push` the local repo to the remote one on GitHub:

```
git remote add origin git@github.com:your_username/your_repo.git
git branch -M main
git push -u origin main
```

- Need to do a `push` every time you want your local changes to be reflected on GitHub.

# Git, visualized



# Basic git commands

- `git init` - create a new git repository in the current directory
- `git status` - "what's currently going on in this repository?"
  - What's our branch?
  - List the commits
  - List the changed files
- `git add <filename>` - add a file to the staging area
- `git rm --cached <filename>` - remove a file from the staging area
- `git commit` - commit the changes in the staging area
  - `git commit -m "message"` - commit with a **message** explaining what the purpose of this new "version" is
- `git log` - show the history of commits

# Pushing and Pulling

- `git push` - send your local changes to the remote repository
- `git pull` - get the latest changes from the remote repository




# Creating a private Github repo for HWs (optional)



- Create an account on github.com
- Click “Your Repositories” → “New”

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)



 [Single sign-on](#) to see more options within the **upenn** organization.

**Owner \*** **Repository name \***

 ngernest ▾ / cis2400\_hw1\_demo 

Great repository names are short and memorable. Need inspiration? How about [refactored-octo-guacamole](#)?

### Description (optional)

-  **Public**  
Anyone on the internet can see this repository. You choose who can commit.
-  **Private**  
You choose who can see and commit to this repository.

← **Make sure this is set to private!**

# Creating a private Github repo for HWs (optional)

- Navigate to your HW folder on the VM
- Then run the following commands:

```
# Create an empty git repository
git init
```

```
# Make sure we set the private repo on Github as the remote
git remote add origin https://github.com/<github_username>/<repo_name>.git
```

```
# Do some local changes and add them to the Git staging area
git add <file_to_add>
```

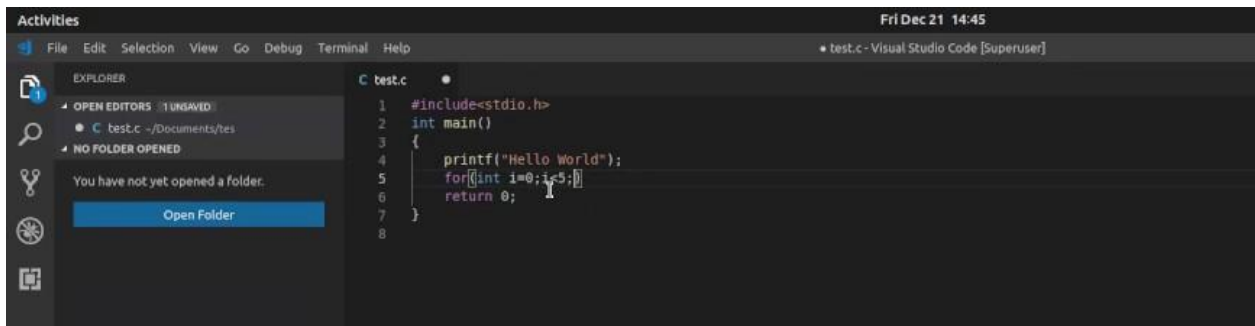
```
# Commit changes
git commit -m "commit_message_goes_here"
```

```
# Call the current branch "main"
git branch -M main
```

```
# Push local changes to the remote (Github)
git push -u origin main
```

# What is VS Code?

- VS Code is a free, lightweight and powerful code editor (developed by Microsoft)
- Easy to use interface, many extensions for coding in different languages
- The CIS 2400 VM comes pre-installed with Sublime Text (another code editor)
  - The next few slides will show you how to install VS Code on the VM if you want (this is optional)



# Downloading VS Code on Ubuntu (optional)

Download from this link:

<https://code.visualstudio.com/docs/setup/linux>

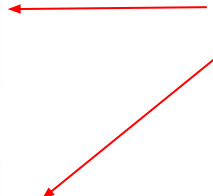
Installing the .deb package will automatically install the apt repository and signing key to enable auto-updating using the system's package manager. Alternatively, the repository and key can also be installed manually with the following script:

```
sudo apt-get install wget gpg
wget -qO- https://packages.microsoft.com/keys/microsoft.asc | gpg --dearmor > packages.micr
oft.gpg
sudo install -D -o root -g root -m 644 packages.microsoft.gpg /etc/apt/keyrings/packages.micr
osoft.gpg
sudo sh -c 'echo "deb [arch=amd64,arm64,armhf signed-by=/etc/apt/keyrings/packages.microsoft.
gpg] https://packages.microsoft.com/repos/code stable main" > /etc/apt/sources.list.d/vscode.
list'
rm -f packages.microsoft.gpg
```

Then update the package cache and install the package using:

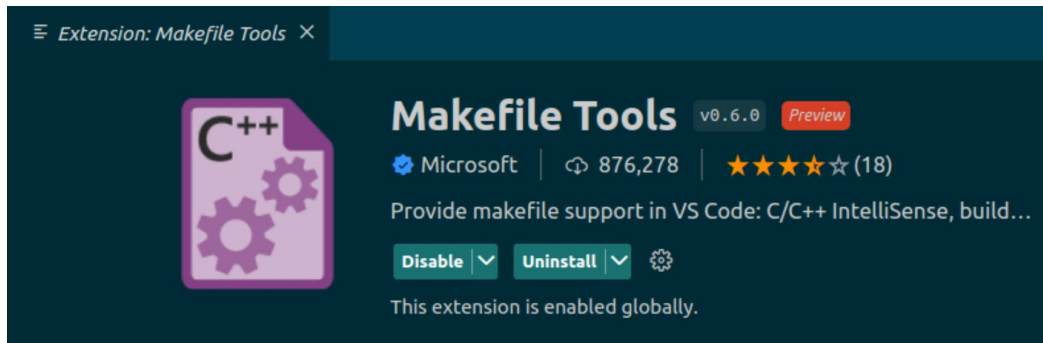
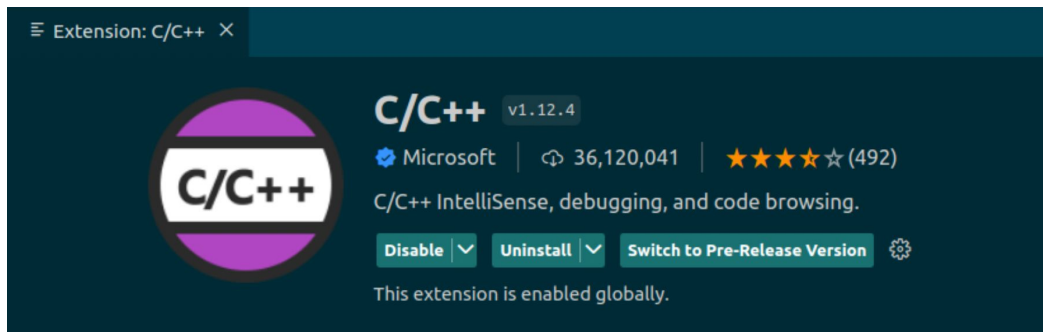
```
sudo apt install apt-transport-https
sudo apt update
sudo apt install code # or code-insiders
```

Follow the  
instructions here!



# Installing the C & Makefile Extensions for VS Code (optional)

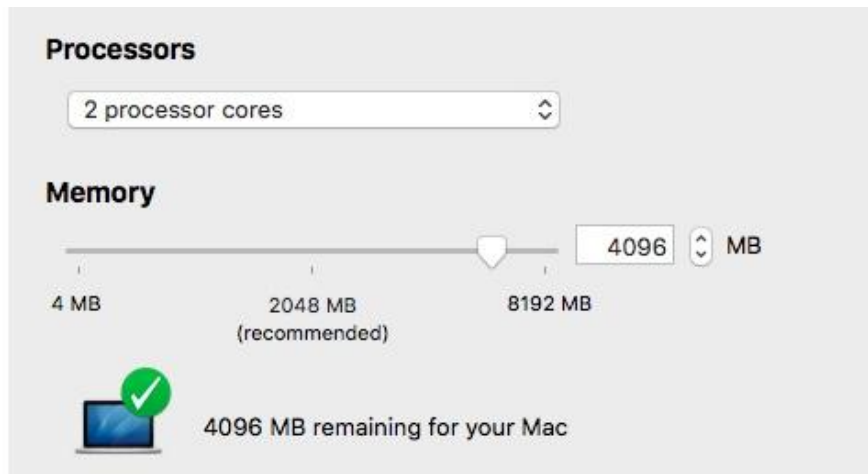
In VS Code: Control + Shift + P → “Extensions: Install Extensions” → search for “C” & “Makefile” respectively



These extensions provide syntax highlighting and other quality-of-life Improvements when you’re doing C programming in VS Code

## Increasing the memory on your VM (optional)

- Make sure the VM is not turned on when you do this!
- On a Mac: Click “Virtual Machine” in the toolbar → “Settings” → “Processors & Memory”
- Increase the virtual memory to something more than 2 GB (eg. 4 GB, i.e. 4096 MB)
  - If you don’t know how much RAM your laptop has, 4 GB should suffice for the VM



# Compiling & Testing HW1

- On the terminal, navigate to the folder containing your `bits.c` file
- Run the command `make` in the terminal to compile your C code
  - Run this command before anything else!
- `btest`: Checks the correctness of the functions in `bits.c`
  - Type the following command into the terminal: `./btest`
  - To test only a single function: `./btest -f bitAnd`
- `dlc`: Check for compliance with the coding rules for each puzzle
  - Run the command: `./dlc bits.c`
  - Program runs silently unless it detects a problem (eg. too many operators)
  - Print the no. of operators used by each function: `./dlc -e bits.c`
- `fshow`: See what an arbitrary bit pattern represents as a floating-point number

# That's all we have for today!

## Reminders:

- TA-lead recitations will take place on
  - Tuesdays 6:30-8:00pm in Moore 100A
  - Wednesday 12:00-1:30pm in Moore 100C
- Check the course website for OH times
- HW1 is due this Friday, 9/23 at 11:59pm