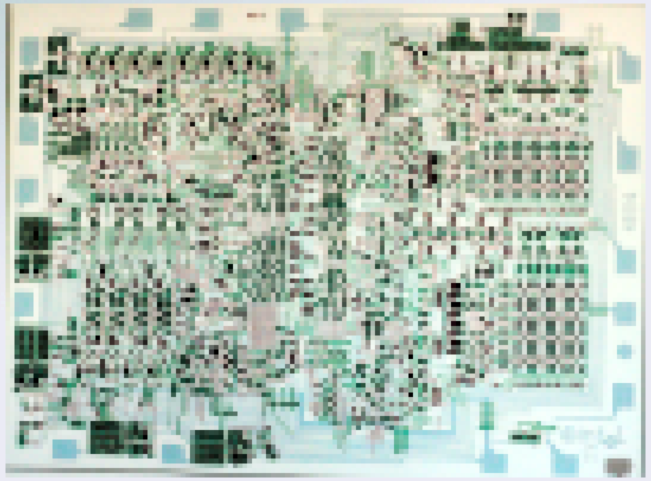

Lecture 1b: Tools

CSCI11: **Computer Architecture and Organization**

Common Tools Used in Software Development

- **Integrated Development Environments (IDEs)** — Combine editors, debuggers, and automation capabilities
    - **Visual Studio Code**, **NetBeans**, **Sublime Text**, **Cursor**
- **Distributed version control system** - **Git**, *GitHub*, *GitLab*, *Bitbucket*
- **Build & Automation Tools** - *Jenkins*, *Automated build systems*
- **Testing & Quality Assurance** - *Testing frameworks*, *Browser DevTools*
- **Deployment & Infrastructure** - *Docker*, *VMWare Workstation*
- **Development Support** - *Debuggers*, *Frameworks & Libraries*, *Software Development Kits (SDKs)*

Tools Required for This Class

- `git` running on your school computer (git (macOS) or git bash (Windows))
- `code editor` running on your computer (*VS Code*)
- `terminal` program running on your computer (*VS Code*)
- `markdown` as **the** way to format text
- *github.com* with an SSH login, an instructor repository, and a student repository

All of these tools are in heavy use by professionals in technical fields, it is important for you to know how to use them

Part 1: Git

What is Git?

Git is a **distributed version control system (VCS)** used to track changes in source code during software development.

- Tracks file changes over time
- Enables collaboration across teams
- Maintains complete project history
- Allows reverting to previous versions when needed

Why Git Matters

- **Industry Standard**: Used by professionals worldwide
- **Resilient Design**: Every developer has a full copy of the repository, eliminating single points of failure
- **Collaboration**: Multiple developers can work on the same project simultaneously
- **Career Essential**: Required knowledge for software development roles

Git Architecture

Git works by:

1. Storing complete project history locally
2. Using a distributed model (not dependent on central server)
3. Creating snapshots of your code at specific points
4. Allowing developers to work offline and sync later

Basic Git Workflow

```
Working Directory → Staging Area → Local Repository → Remote Repository
    (Your files)      (git add)      (git commit)       (git push)
  Perform edits.....stage them....save locally.........save on remote
```

Essential Git Commands

- `git init` — Initialize a new repository (*do only once*)
- `git add -A` — **Stage** changes for commit
- `git commit -m "message"` — **Commit** changes to history
- `git push` — **Push** or send changes to remote repository
- `git pull` — Pull or retrieve changes from remote

Git Repositories

A Git repository (or **repo**) is a folder that stores:

- All project files
- Complete history of changes
- Version information

Two main types:

- **Local**: Stored on your computer
- **Remote**: Hosted on servers like GitHub

Part 2: VS Code

What is VS Code?

Visual Studio Code is a **lightweight, open-source code editor** developed by Microsoft.

- Works on Windows, macOS, and Linux
- Supports multiple programming languages
- Extensible with thousands of plugins
- Industry standard among developers

Why VS Code is Important

Current Industry Adoption: Over 70% of developers use VS Code according to Stack Overflow surveys

Benefits:

- **Frictionless**: Gets out of your way so you can focus on coding
- **Extensible**: Customize with extensions for your needs
- **Integrated**: Built-in terminal, git control, and debugging
- **Cross-Platform**: Same experience on any operating system

VS Code Features for Architecture Course

- **Integrated Terminal**: Run commands without leaving the editor
- **Git Integration**: View changes, stage commits directly
- **Markdown Preview**: See formatted documents in real-time
- **Syntax Highlighting**: Better code readability
- **Extensions**: Add language support as needed

Getting Started with VS Code

1. Download from code.visualstudio.com
2. Install for your operating system
3. Open your project folder
4. Start editing files
5. Use integrated terminal with Ctrl+` (backtick)

Typical VS Code Workspace Setup

```
Your Project Folder
├── README.md
├── .gitignore
├── src/
│   ├── file1.c
│   └── file2.s
└── .git/
```

Open the **project folder in VS Code** to maintain organization and leverage Git integration.

DO NOT OPEN SINGLE FILES IN VS CODE

Part 3: Terminal

What is a Terminal?

The terminal — also called the **command line**, **shell**, or **console** — is a **text-based interface** that lets you interact directly with your computer's operating system.

- Execute commands by typing text
- No graphical interface required
- Powerful and precise control

Why Terminal Skills Matter

- **Efficiency**: Faster than using graphical menus for many tasks
- **Professional Standard**: Used by developers, system administrators, and engineers
- **Git Operations**: Execute Git commands directly
- **System Access**: Compile code, run programs, manage files

Terminal Basics

When you open a terminal, you interact with a **shell** — the software that executes your commands.

The terminal emulates hardware from the 1960s-1980s, but now communicates digitally with your shell rather than via physical wire.

Common Terminal Commands

```
pwd                # Print working directory
ls                 # List directory contents
cd folder_name     # Change directory
mkdir new_folder # Create new folder
git status         # Check Git status
```

Terminal in VS Code

Integrated Terminal: Open inside VS Code without leaving the editor

- Keyboard shortcut: *Ctrl* + ` (backtick, left of "1" key)
- Right-click → "Open in Integrated Terminal"
- Multiple terminals: Click `+` icon to add tabs

Benefits:

- Edit code and run commands in same window
- View output immediately
- Maintain context of your project

Part 4: Markdown

What is Markdown?

Markdown is a **lightweight markup language** for formatting text using simple, readable syntax.

- Uses plain text characters for formatting
- Converts to HTML, PDF, and other formats
- Git-friendly and documentation standard

Why Markdown is Essential

- **Lightweight and Fast**: Runs on any plain-text editor
- **Cross-Platform**: Works on Windows, macOS, Linux, iOS, Android
- **Git-Friendly**: Standard for GitHub README files and documentation
- **Professional Standard**: Used for technical documentation across the industry

Markdown Formatting Basics

Element	Syntax	Example
Heading	`# Text`	# Heading
Bold	`**text**`	**bold**
Italic	`*text*`	*italic*
Code	`` `code` ``	`inline code`
Link	`[text](url)`	GitHub

Markdown Code Blocks

Markdown supports code blocks with syntax highlighting:

~~~
```python
def hello():
    print("Hello, World!")
```
~~~

Specify the language for proper highlighting (python, c, javascript, etc.)

Markdown Document Structure

Use heading hierarchy to organize content:

Code:

```
## Main Title
## Major Section
### Subsection
#### Detail Level

- Bullet point 1
- Bullet point 2
```

Creates professional, readable documentation.

Markdown Document Structure

Use heading hierarchy to organize content:

Preview:

Main Title

Major Section

Subsection

Detail Level

- Bullet point 1
- Bullet point 2

Using Markdown in Your Course

- **README files**: Explain your projects
- **Comments**: Document code functionality
- **Notes**: Organize study materials
- **Presentations**: Create slides with Marp (like this one!)
- **GitHub**: Write descriptions and documentation

Part 5: GitHub SSH Authentication

What is GitHub.com?

GitHub is a cloud platform that hosts **remote Git repositories**, enabling:

- Centralized code storage
- Team collaboration
- Code sharing and version control
- Community contribution

What is SSH?

SSH (**Secure Shell Protocol**) provides a **secure channel over unsecured networks**.

For GitHub:

- Creates secure, encrypted connection
- Uses public and private key pairs
- More secure than password authentication
- Industry standard for Git operations

SSH Key Pair Basics

An SSH key pair consists of:

- **Public Key**: Safe to share; placed on GitHub
- **Private Key**: Keep secret; stored locally on your computer

The pair works together to authenticate you securely without sending passwords.

Setting Up GitHub SSH (Overview)

1. **Generate SSH key pair** on your computer
2. **Check for existing keys** to avoid duplicates
3. **Add public key to GitHub account**
4. **Test SSH connection** to verify setup

Detailed directions are in the student repository.

GitHub Student Repository

For this course:

- You'll receive a **student repository** link
- Clone it using SSH for secure, password-free access
- Make changes locally using Git
- Push changes back to GitHub
- Private with Instructor as a collaborator

Benefits:

- Assignments tracked in version control
- Collaboration with instructor
- Professional workflow experience

GitHub Instructor Repository

For this course:

- You'll receive a **instructor repository** link
- Clone it using SSH for secure, password-free access
- Pull to get weekly assignments
- Copy assignments to student repo to become homework

Benefits:

- Assignments tracked in version control
- Collaboration with instructor
- Professional workflow experience

Git Clone with SSH

```
git clone git@github.com:username/repo-name.git
cd repo-name
```

This creates a local copy of your remote repository, ready to work with.

Homework Process

1. Pull latest assignments

```
cd CSCI11_Sp26_Instructor
git pull
cp digital_design/week_1/ ../CSCI11_Sp26_Student/week_1

## edit/code/work on assignment

git add -A
git commit -m "homework week 1"
git push
```

How These Tools Work Together

1. **VS Code Editor** — Edit your code
2. **Markdown** — Document your work
3. **VS Code Terminal** — Execute CLI commands
4. **Git** — Track changes locally
5. **GitHub SSH** — Push changes securely to remote

Professional Development Skills

> "All of these tools are in heavy use by professionals in technical fields, it is important for you to know how to use them."

Mastering these tools:

- Makes you job-ready
- Enables efficient collaboration
- Establishes professional practices
- Builds foundation for advanced topics

Summary: What You've Learned

- **Git**: Version control for tracking code changes
- **VS Code**: Industry-standard code editor
- **Terminal**: Powerful command-line interface
- **Markdown**: Professional documentation formatting
- **GitHub SSH**: Secure remote repository access

All essential for success in computer architecture and beyond.

Next Steps: Overview

1. Install/confirm VS Code on your computer
2. Install/confirm Git (or Git Bash on Windows) on your computer
3. Generate and configure SSH keys for GitHub
4. Clone your student repository
5. Fix the README in your repository
6. You will gain access to the Instructor Repo next week