PSAM 5600 Section E CRN 4598
Spring Semester 2017
Wednesdays 7:00PM – 9:40PM

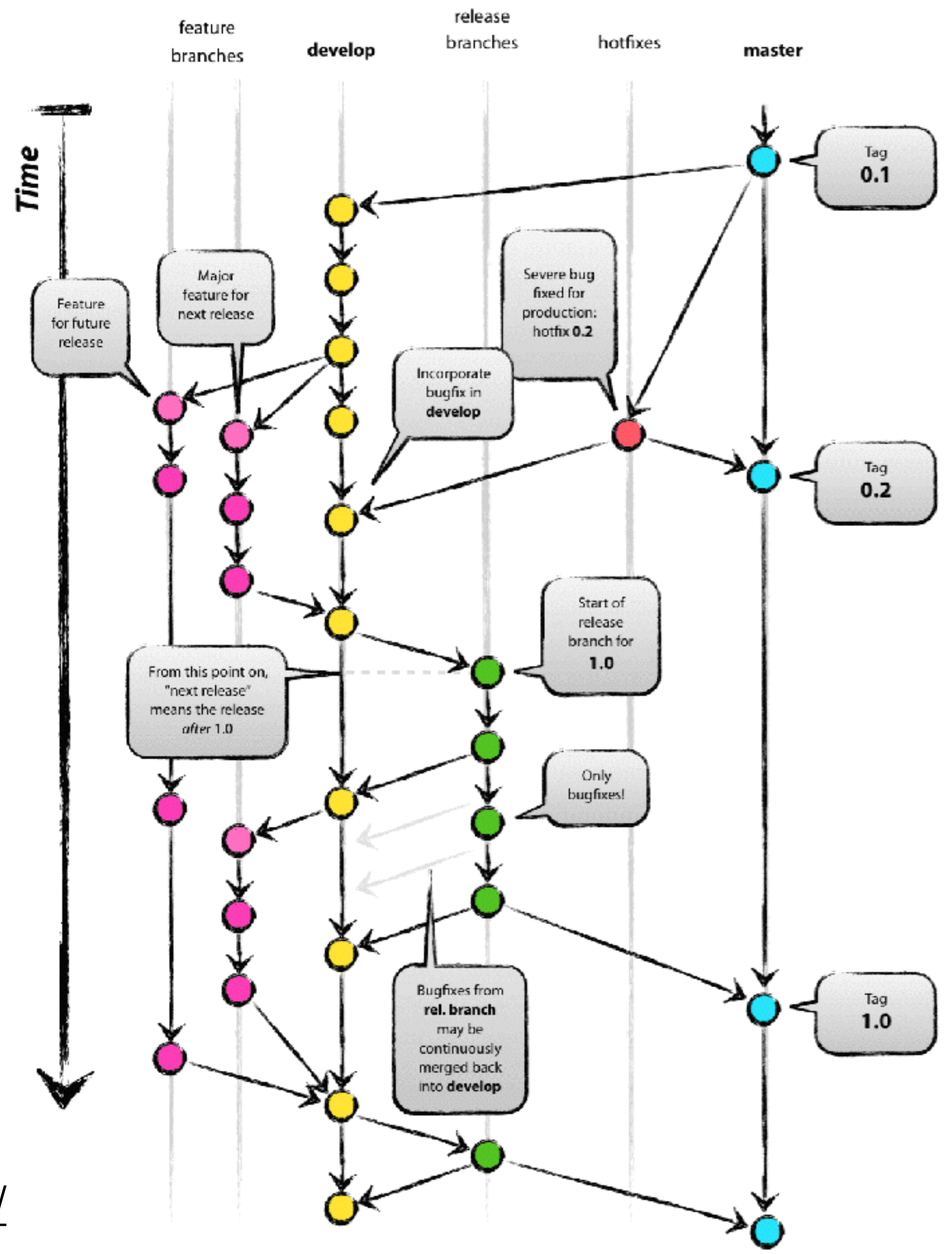
Currents: Coding with Cinder

Week 5: Video & Audio Playback / Building & Debugging a Program

Instructors
Luobin Wang (luobin@newschool.edu)
Weili Shi (weili@newschool.edu)

EXAM

Create a smart pointer class.



Version Control with git-flow

Git-flow cheatsheet: http://danielkummer.github.io/git-flow-cheatsheet/

http://nvie.com/posts/a-successful-git-branching-model/

Task 2: Collaborate with your fellow programmer!

12. **Both:** Pull the newest code from GitHub:
    git pull
13. **Member 1:** create a feature branch and work on it:
    git flow feature start **answer1**
    Modify the value of the answer variable to **43**; commit and publish:
    git add -A
    git commit -m “Modify answer to 43"
    git flow feature publish **answer1**
14. **Member 2:** create a feature branch and work on it:
    git flow feature start **answer2**
    Modify the value of the answer variable to **45**; commit and publish:
    git add -A
    git commit -m “Modify answer to 45"
    git flow feature publish **answer2**

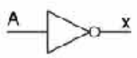
Task 2: Collaborate with your fellow programmer!

15. **Member 1:** On repo website, create a pull request for the **feature/answer1** branch; merge it into **develop** branch; and delete the feature branch. In terminal, Finish (delete) the local feature branch:
    git checkout develop
    git pull
    git flow feature finish **answer1**
16. **Member 2:** On repo website, create a pull request for merging **feature/answer2** into **develop**.
17. **Both:** Now we see a conflict. All conflicts must be resolved before merging. Both team members should sit together, click “Resolve conflicts” button, look at the conflicting parts in the code, decide which part to keep/delete, resolve all conflicts, commit changes, merge the pull request, and finally delete the feature branch.
18. **Member 2:** Finish (delete) the local feature branch:
    git checkout develop
    git pull
    git flow feature finish **answer2**
19. **Member 1:** Update the repo according to GitHub:
    git checkout develop
    git pull

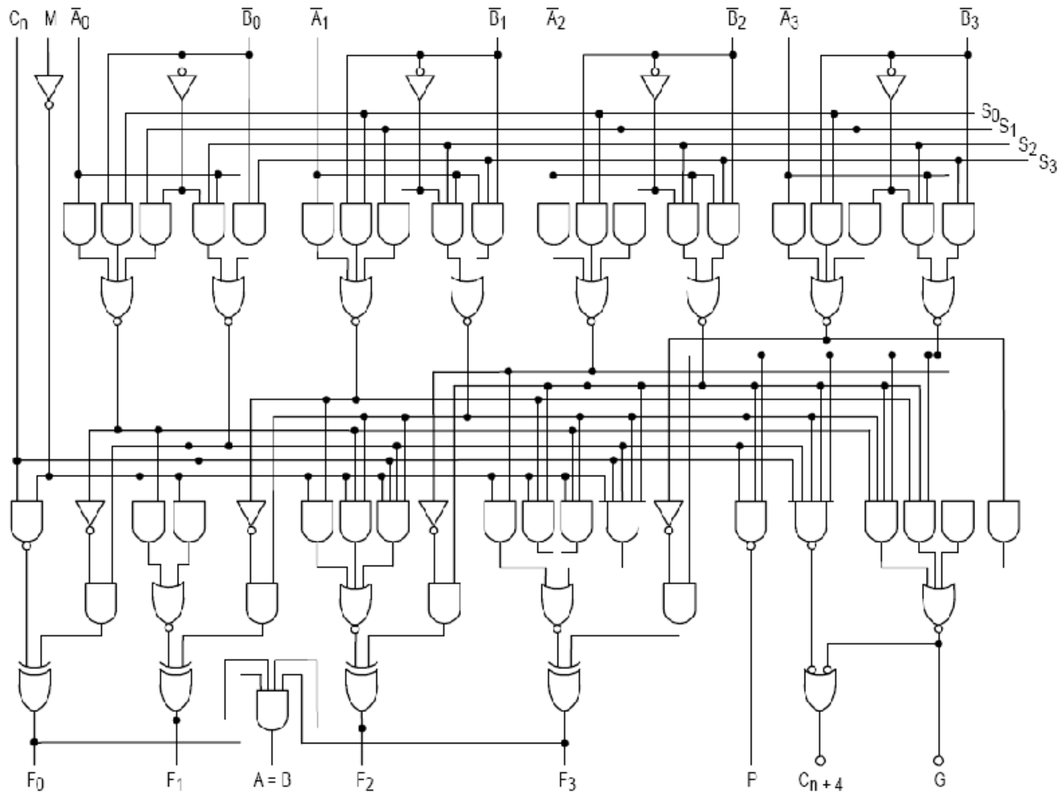
Task 3: Clone your homework repo into Cinder root folder

1. In terminal, go to your git repo, commit all changes, push to GitHub:
   cd **<path to your repo>**
   git commit -am **"Update files."**
   git push
2. Delete your local repo:
   cd ..
   rm -rf **<path to your repo>**
3. Go to your repo website, copy the HTTPS link to your repo.
4. In terminal, go to your Cinder root folder, clone your repo there:
   cd **<path to your Cinder root>**
   git clone **<HTTPS link to your repo>**
5. From now on, when doing homework, use TinderBox to create Cinder projects within your repo directory. In the future, if you need to clone your repo to another computer, clone it into Cinder root folder, too. Your projects (since this week) should be able to build and run on the new computer as well. (We'll try to fix your earlier projects in the next week!)

Logic Gates

Name	NOT	AND	NAND	OR	NOR	XOR	XNOR
Alg. Expr.	$\overline{A}$	AB	$\overline{AB}$	$A+B$	$\overline{A+B}$	$A \oplus B$	$\overline{A \oplus B}$
Symbol							

Truth Table																	
A	X	B	A	X	B	A	X	B	A	X	B	A	X	B	A	X	B	A	X
0	1	0	0	**0**	0	0	**1**	0	0	**0**	0	0	**1**	0	0	**0**	0	0	1
1	**0**	0	1	**0**	0	1	**1**	0	1	**1**	0	1	**0**	0	1	**1**	0	1	**0**
		1	0	**0**	1	0	**1**	1	0	**1**	1	0	**0**	1	0	**1**	1	0	**0**
		1	1	**1**	1	1	**0**	1	1	**1**	1	1	**0**	1	1	**0**	1	1	**1**



What is a CPU made from?

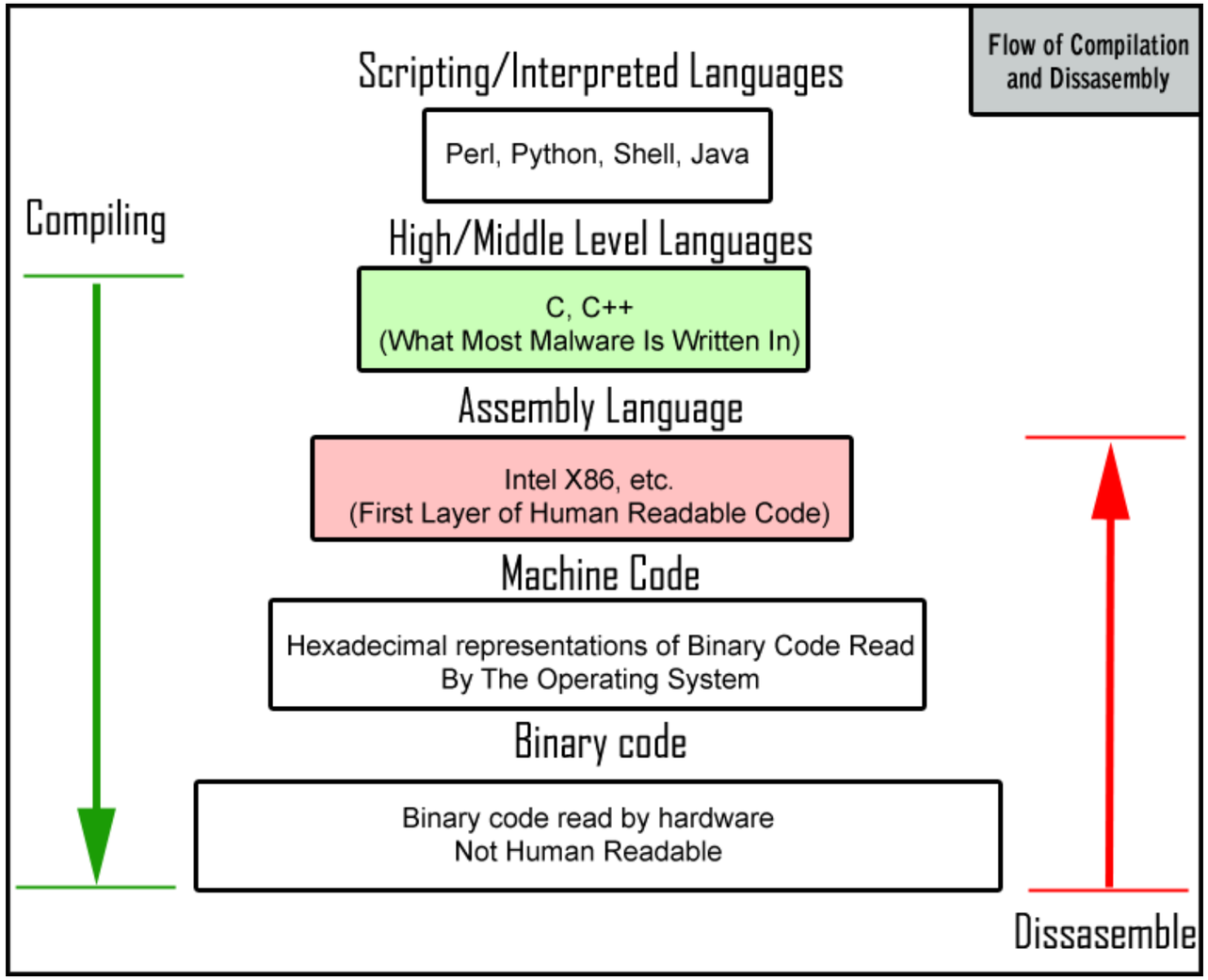
Not only data, but also control signals are represented by 0s and 1s (machine language)!

http://imgur.com/M59IOZQ http://www.wikiwand.com/en/Arithmetic_logic_unit

```
# Equivalent to: SUM = NUM1 + NUM2

MOV AL, NUM1
MOV BL, NUM2
ADD AL, BL
MOV SUM, AL
```

Assembly Language

Human-readable, but still machine-oriented.

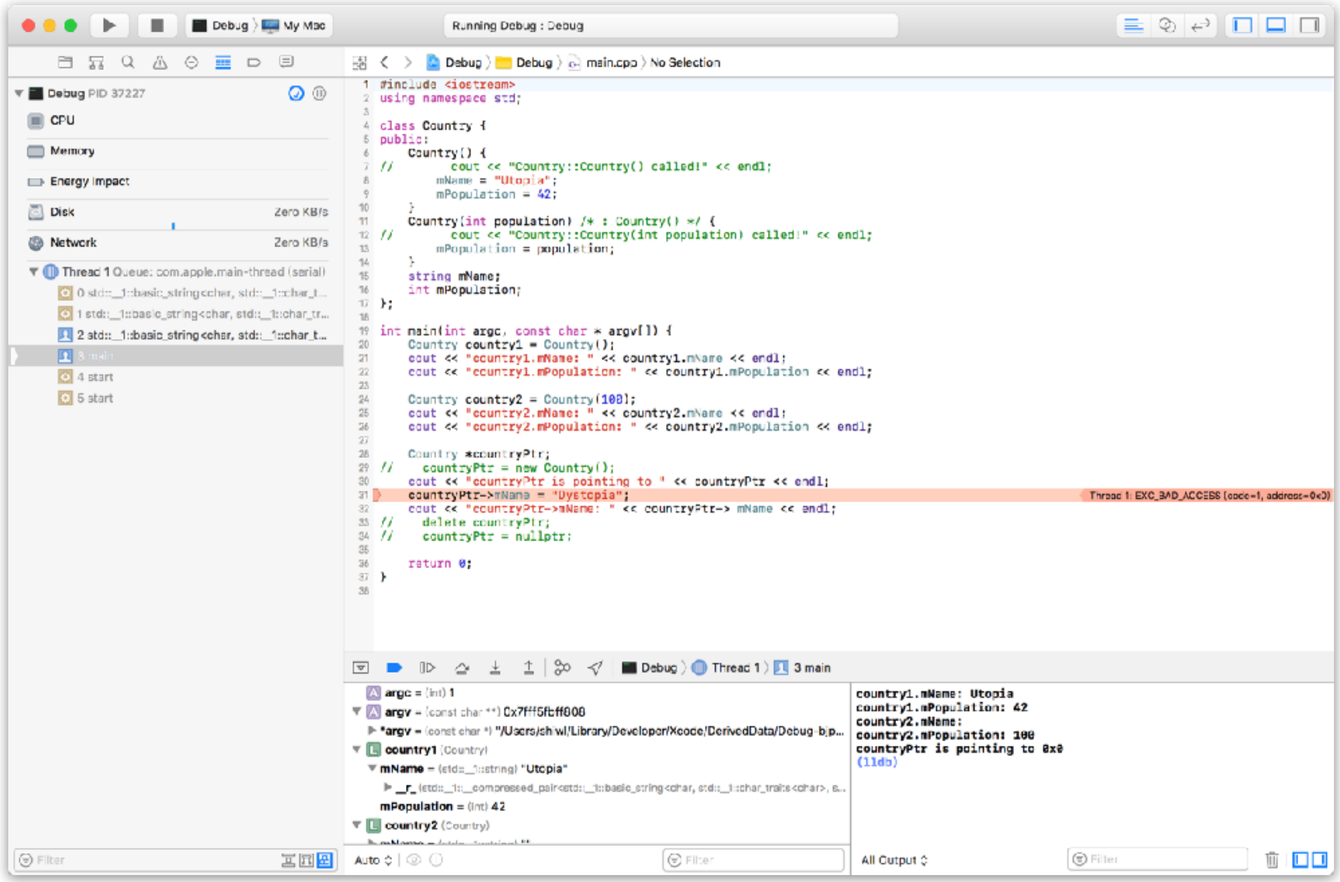


Hierarchy of Computer Languages

https://blog.malwarebytes.com/wp-content/uploads/2012/09/FlowDiagram2.png

```
# preprocess:
clang++ main.cpp -E

# assemble:
clang++ main.cpp -S

# compile:
clang++ main.cpp -c
clang++ function.cpp -c

# link:
clang++ main.o function.o -o main

# run:
./main
```

Building a Program



2 Common Ways to Debug a Program

- Output debug information to console;
- Using the debugger to step through the code.

Cinder Signals

Reduce the load in updates.

```
typedef ci::signals::Signal<void()>
VideoEndedSignal;
```

Declare signal

```
typedef ci::signals::Signal<void(float number, bool state)> VideoEndedSignal;
```

If you want to pass variables

```
signal.connect(std::bind(&ClassName::onSomethingHappend, this));
```

Connect to signal

```
signal.connect(std::bind(&ClassName::onSomethingHappend, this,
std::placeholder::_1,
std::placeholder::_2));
```

If you want to receive variables being passed on.

When you need a class, build a class.

If you work on the same thing a lot. Consider make that functionality a class.
So you can easy access it later.

Homework (USE SMART POINTER)

1. Build a video player. It must have basic controls. It should be able to open files.
2. (Optional)Hey, your video player looks fancier with a seekbar.
3. Prompt the user if the video player is finished. (USE SIGNAL!)
4. Your video player should be a class. So we can reuse them later.
5. (optional) Incorporate your beautiful UI work in your video.