Embedded Software Development

EECS388 Fall 2022

© Prof. Mohammad Alian

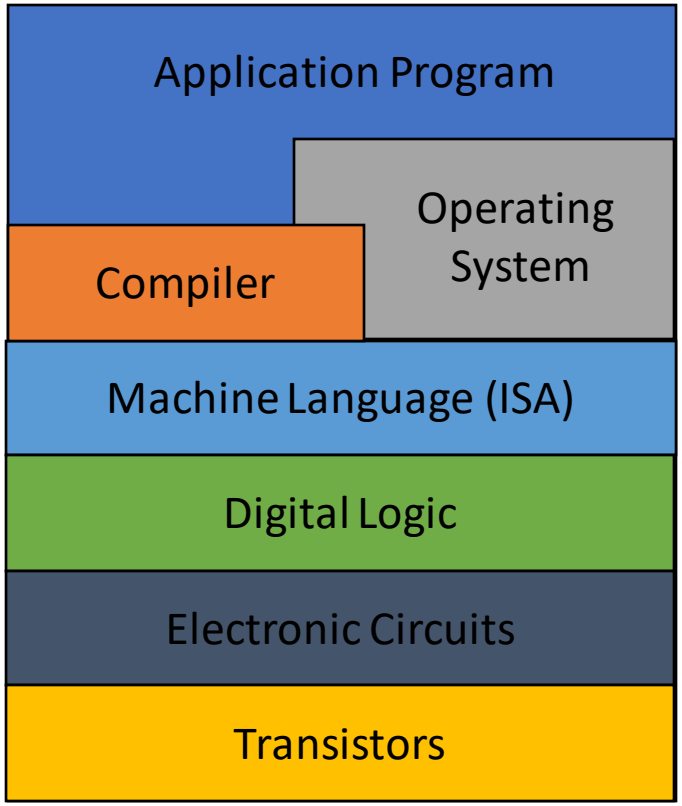
Lecture notes based in part on slides created by Alex Fosdick

Announcement

- Midterm date: Thursday Oct 6th during class time

Context

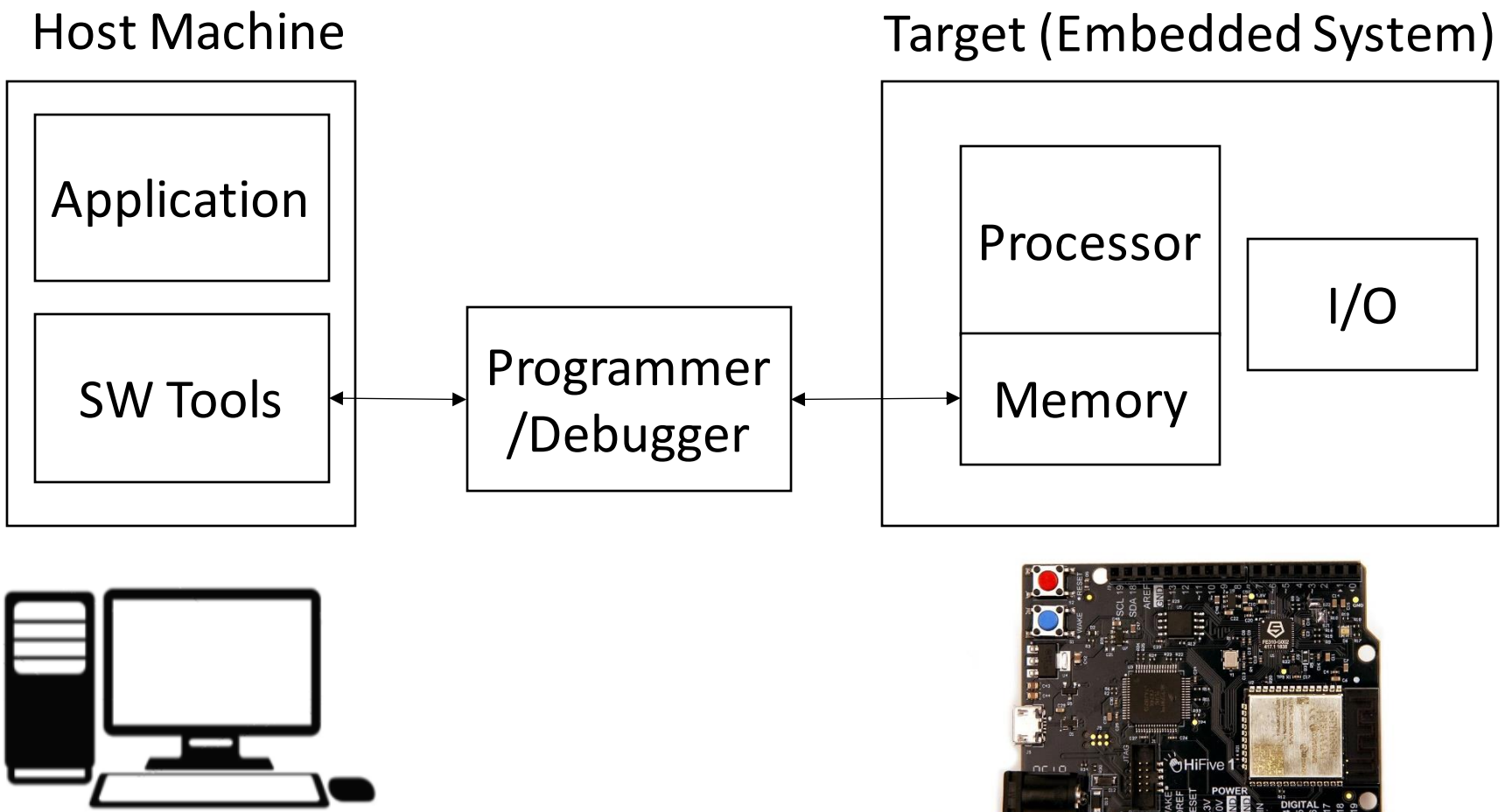
Compilation and software development



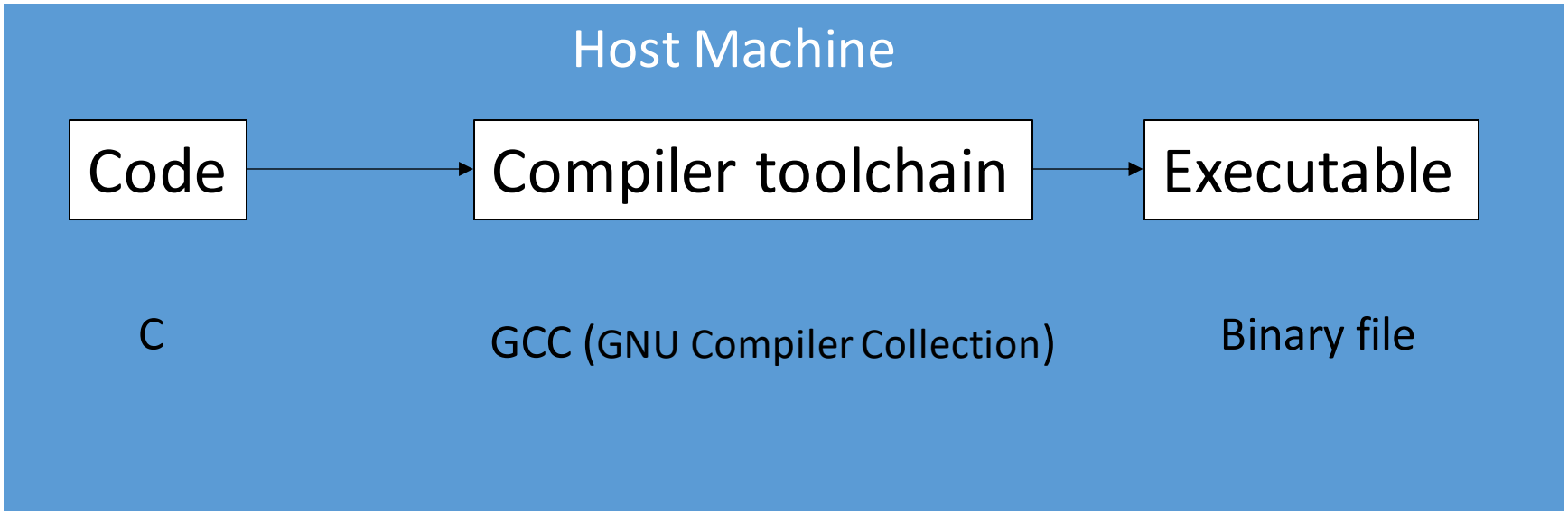
Instruction Set Architecture (ISA)

- Instructions are the “words” of a computer and ISA is its vocabulary
- Defines a standard interface to the processor
- Decouples usage and implementation
  - Intel 486->Pentium->P6->Core Due -> i7
- Examples: x86, ARM, RISC-V, Power, etc.

Embedded System Development Platform

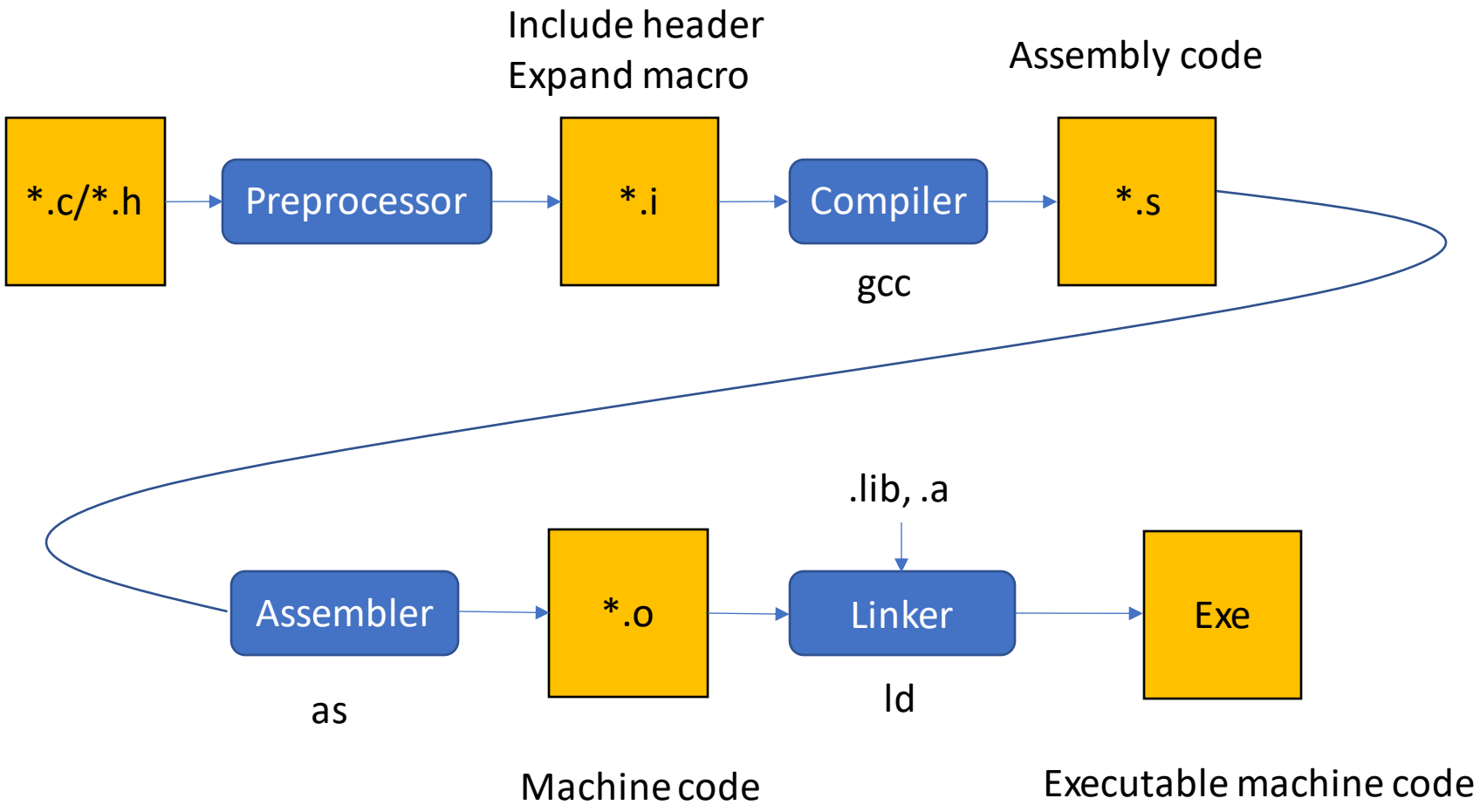


Due to the limited resource of the target, host machine usually contains our build* environment



*Build -> converting a high-level language code to machine executable binary

Compiler Toolchain



C-code —gcc -S→ Assembly —as→ Binary

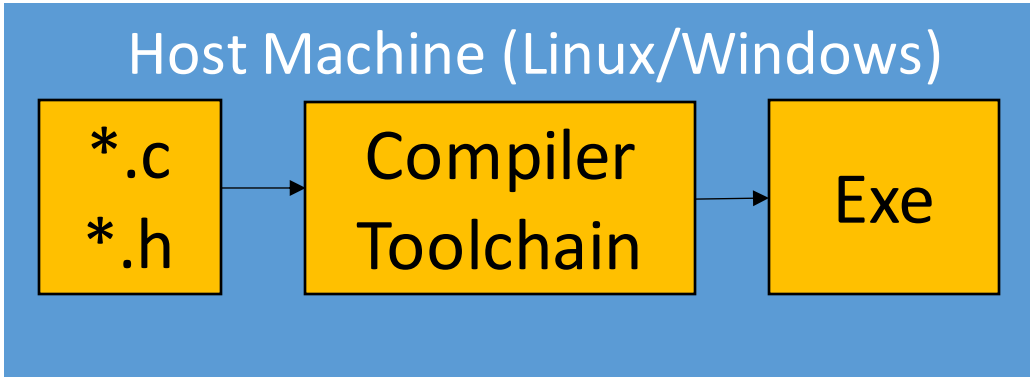
```
int main() {
    int x, y, z;
    x = 5;
    y = 2;
    z = x + y;
    return 0;
}
```

```
 0:   addi    sp,sp,-32
 4:   sd      s0,24(sp)
 8:   addi    s0,sp,32
 c:   li      a5,5
10:   sw      a5,-20(s0)
14:   li      a5,2
18:   sw      a5,-24(s0)
1c:   lw      a4,-20(s0)
20:   lw      a5,-24(s0)
24:   addw    a5,a4,a5
28:   sw      a5,-28(s0)
2c:   li      a5,0
30:   mv      a0,a5
34:   ld      s0,24(sp)
38:   addi    sp,sp,32
3c:   ret
```

```
 0:   fe010113
 4:   00813c23
 8:   02010413
 c:   00500793
10:   fef42623
14:   00200793
18:   fef42423
1c:   fec42703
20:   fe842783
24:   00f707bb
28:   fef42223
2c:   00000793
30:   00078513
34:   01813403
38:   02010113
3c:   00008067
```

Native Compilation

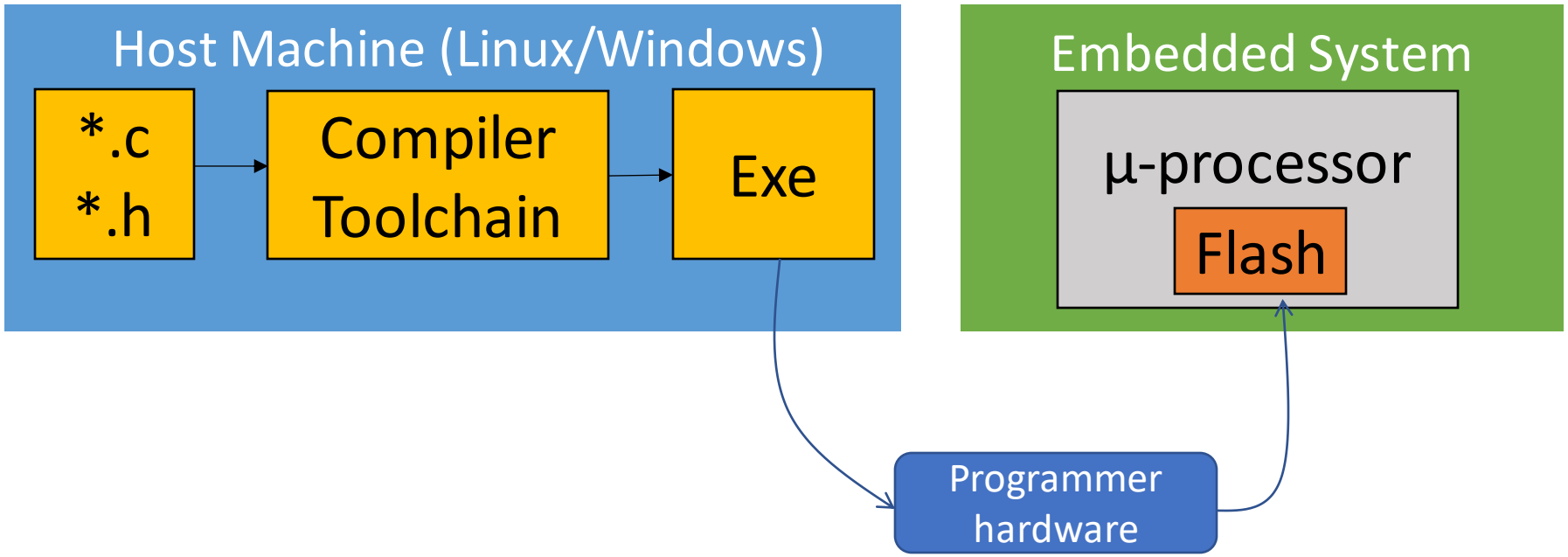
- Compile and run on one system



No hardware needed

Cross Compilation

- Compile on one system and run on another system



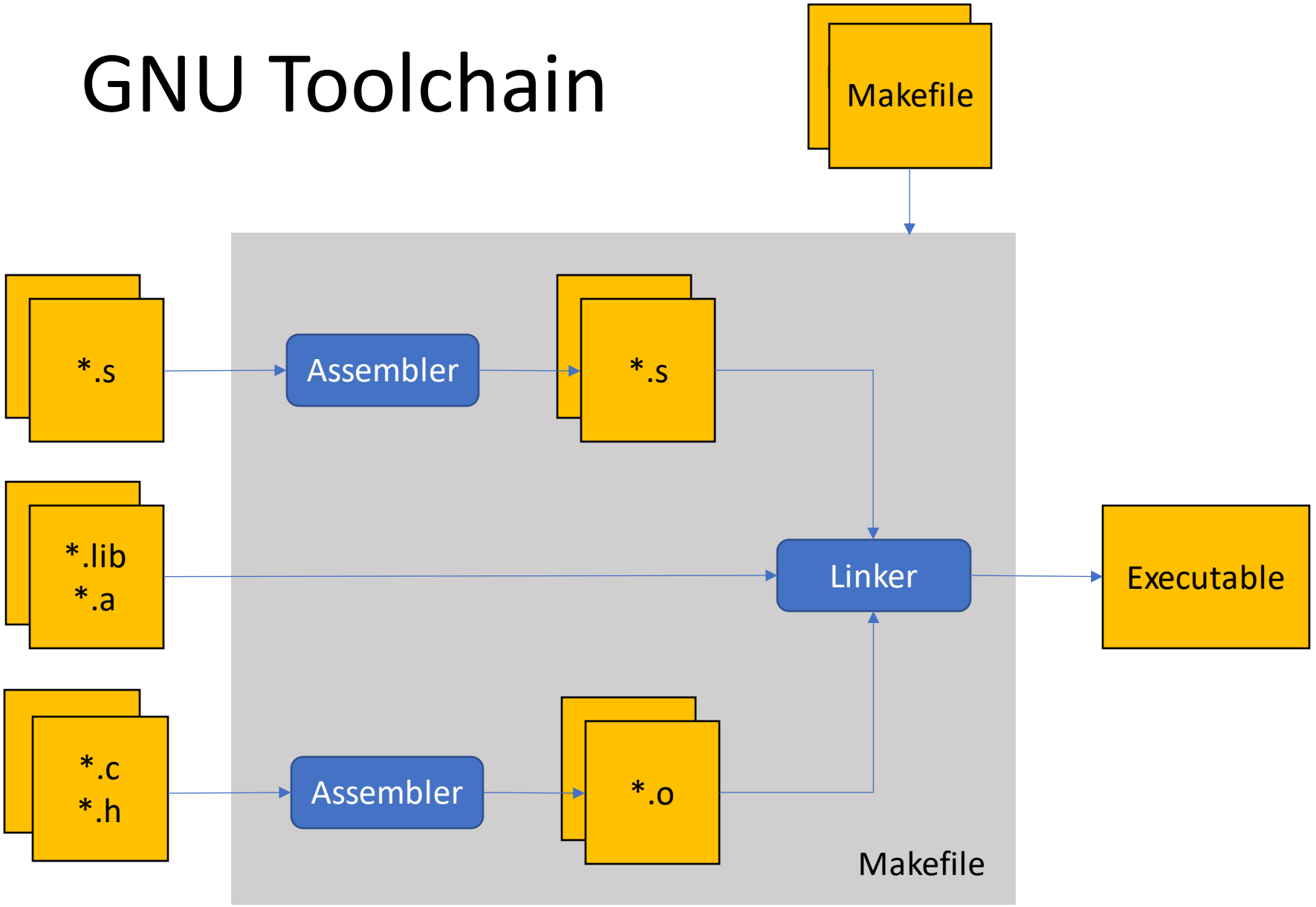
- Building can be too complex
  - Many gcc flags and commands
    - Linux has over 40k+ source code files!
  - Dependencies
  - Many source files
  - Many supported platforms

- Building manually is
  - Not scalable
  - Time consuming
  - Error prone

GNU Make

- “GNU Make is a tool which controls the generation of executables and other non-source files of a program from the program's source files”
  - Preprocessing
  - Compiling
  - Assembling
  - Linking

GNU Toolchain



A Simple Makefile

hellomake.c

```
#include <hellomake.h>

int main() {
  // call a function in another file
  myPrintHelloMake();

  return(0);
}
```

hellofunc.c

```
#include <stdio.h>
#include <hellomake.h>

void myPrintHelloMake(void) {

  printf("Hello makefiles!\n");

  return;
}
```

hellomake.h

```


/*
example include file
*/

void myPrintHelloMake(void);
```

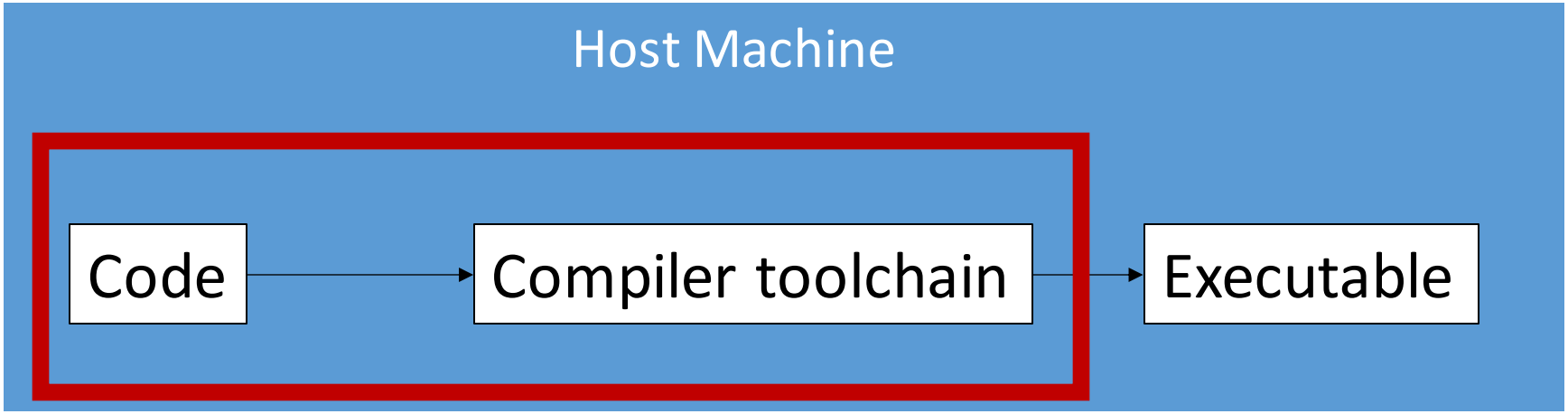
```
CC=gcc
CFLAGS=-I.

hellomake: hellomake.o hellofunc.o
     $(CC) -o hellomake hellomake.o hellofunc.o
```

https://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/

Integrated Development Environment (IDE)

- Autogenerate Makefiles
- Provide a very simple interface for developers (usually beginners)
  - Bad for maintainability and portability
- Professional software teams write their own makefile

- How to keep track of our software changes?
- How to manage to develop complex software using a team of software engineers?

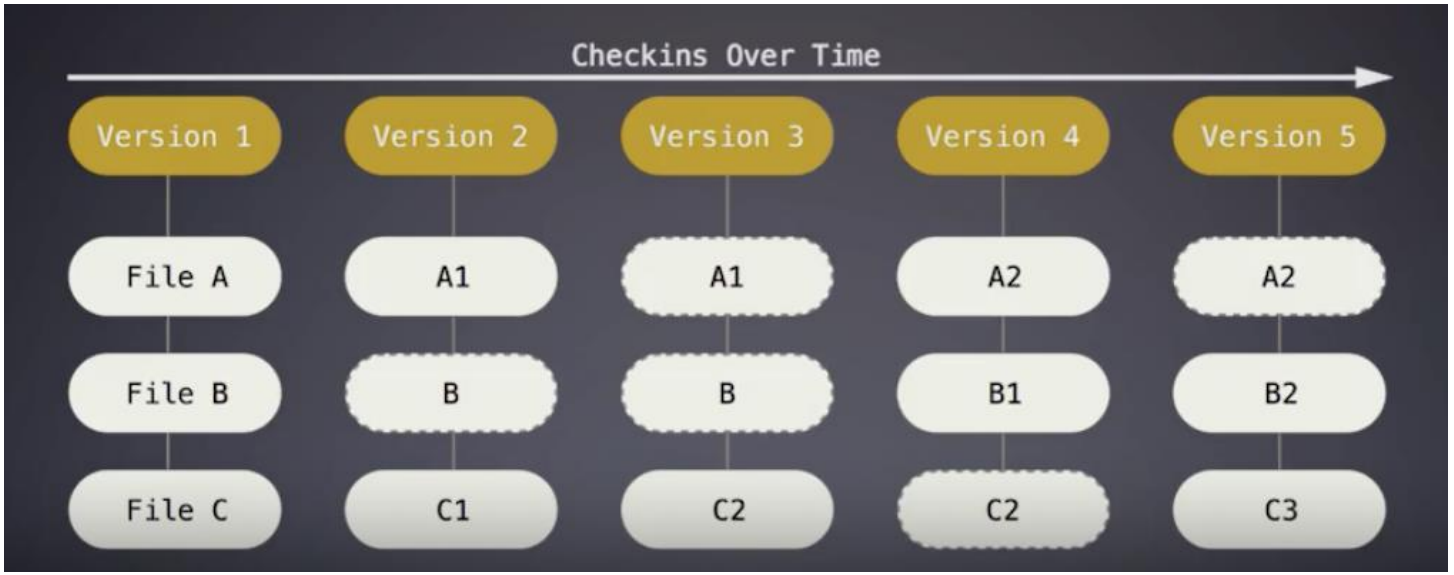


Version Control Systems (VCS)

- A software that allows user to track changes
- Many flavors
  - SVN
  - Mercurial
  - Git

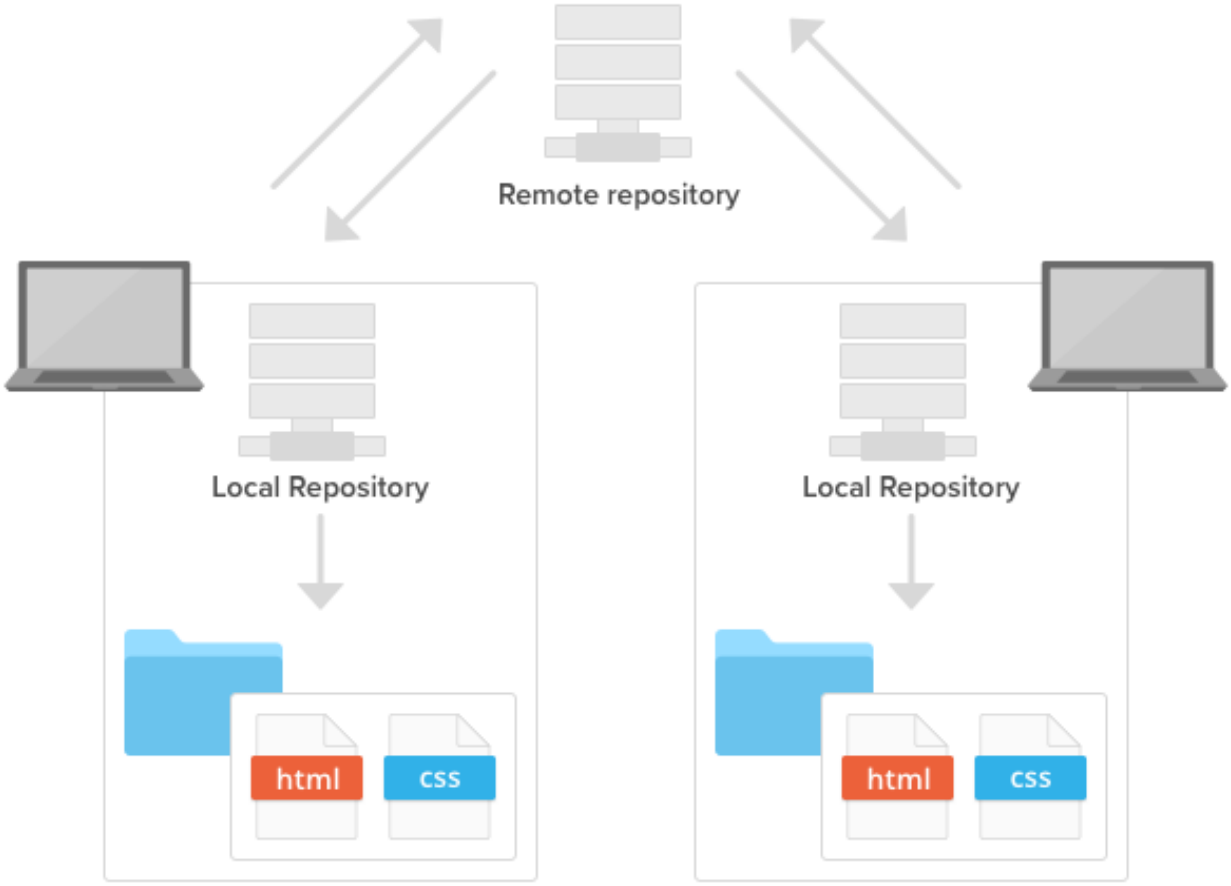
VCS Repository

- Collection of tracked files



https://www.coursera.org/learn/introduction-embedded-systems/lecture/pWGYg/9-version-control

Collaboration with VCSs

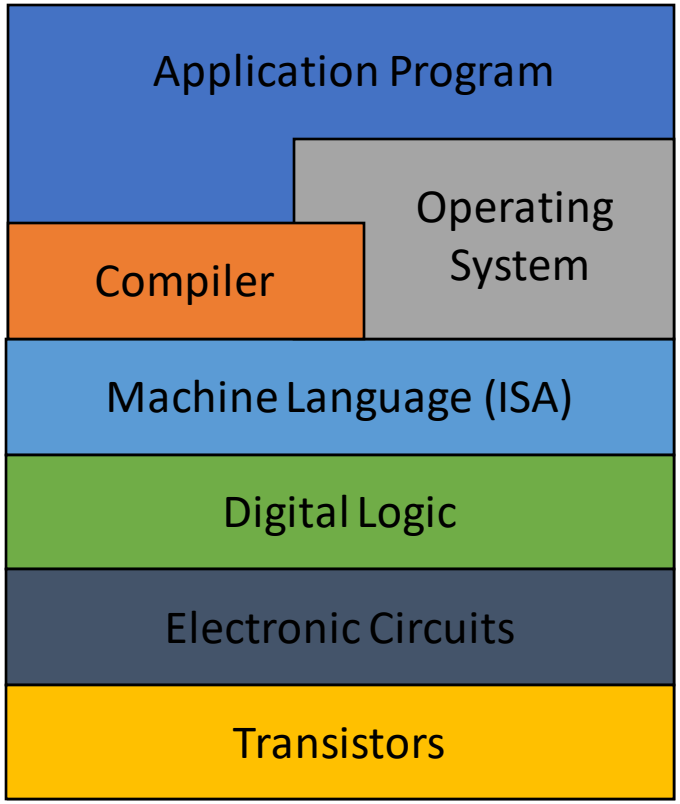


https://backlog.com/git-tutorial/creating-a-repository/

Recap

Prepare on a host machine

Target embedded system



- Compiler toolchain
- Cross compilation
- GNU Make
- IDE
- VCS

What is Next?

- C programming refresher
- Textbook reading
  - There are many resources online for C programming
  - “Introduction to Computing Systems: From Bits & Gates to C & Beyond” Appendix D: The C Programming Language