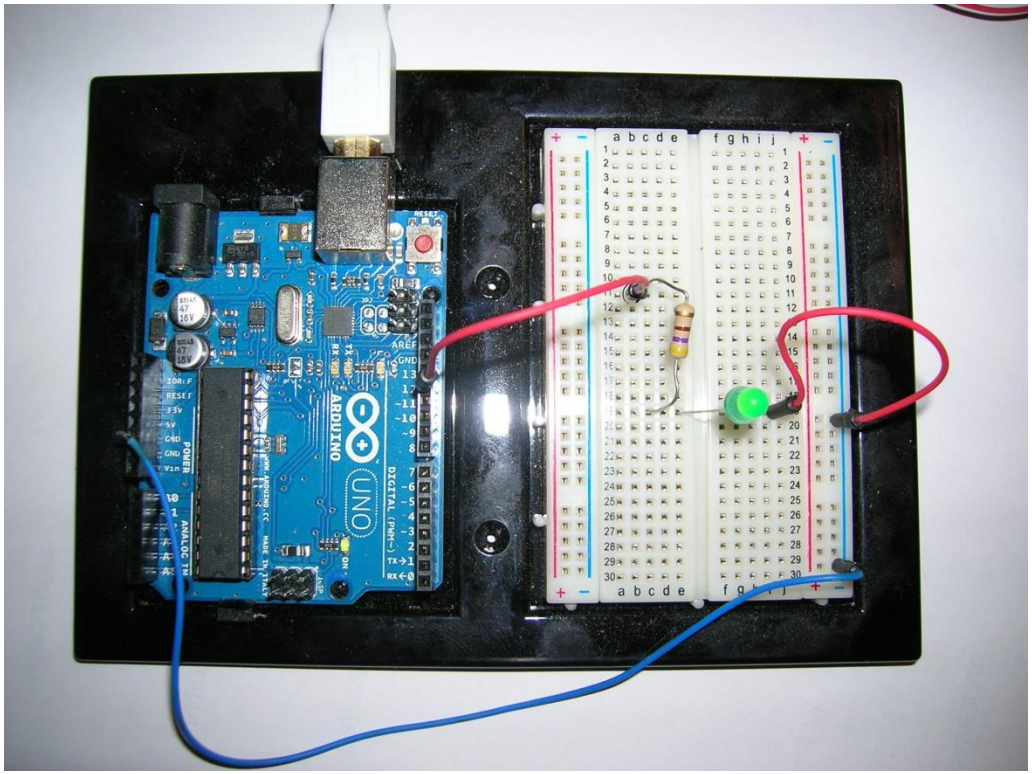
Homework

- Reading Assignment
  - Professional Assembly Language, pp 39-59, 62-65
- Lab 1
  - Download and Read Lab1 Write up
  - Go to lab with your section next week
- MP1
  - Get assignment file from my web page and study it
  - Make and populate your Linux mp1 subdirectory
  - Study the source code for mp1

Embedded Systems Lab

- Introduction to CS341 lab equipment
- Arduino microcomputer system / breadboard

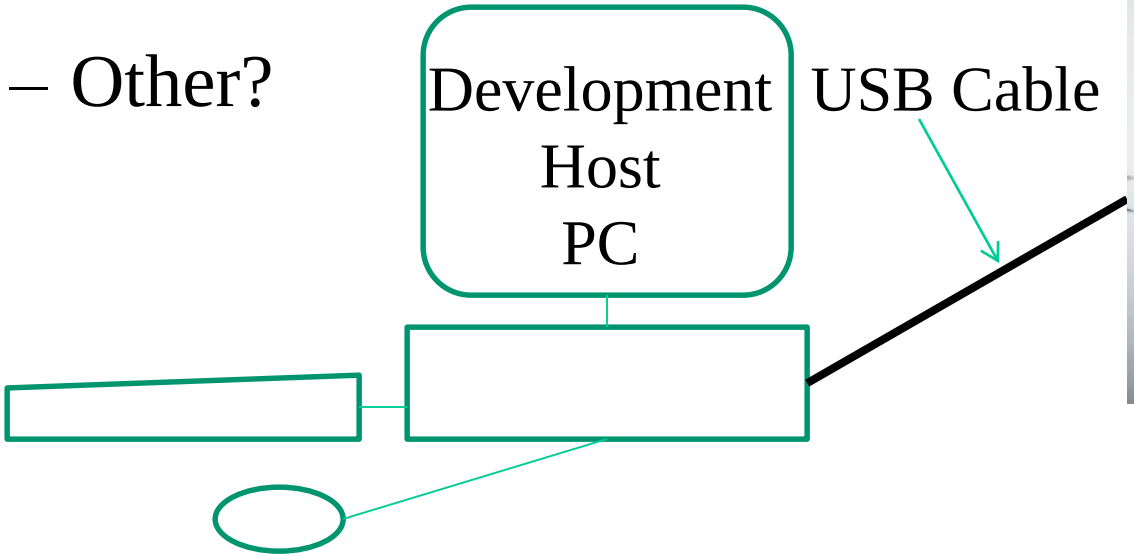


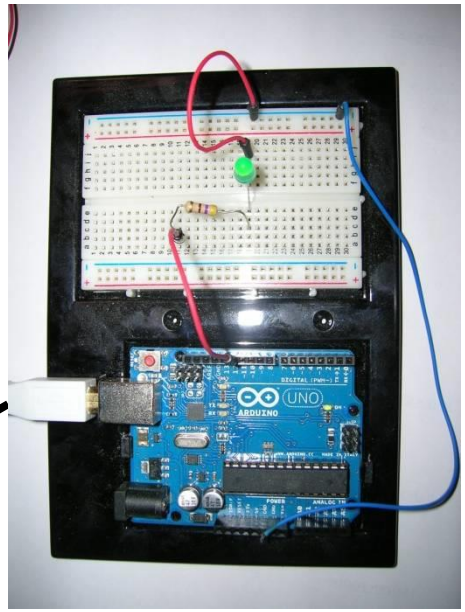
Embedded Systems Lab

- Microprocessor – AtMega328P
  - 8 bit AVR processor @ up to 20 MHz clock
  - 32 Kbytes program memory plus 1 Kbyte RAM
- Programmed using a language like C/C++ with a custom I/O support library
- Development system cost $36.00!
- Buying the processor alone in quantity would probably be about the price of a candy bar

Embedded Systems Lab

- To use embedded software development tools, a development host is attached to a target host via:
  - RS-232
  - Ethernet
  - USB
  - Other?

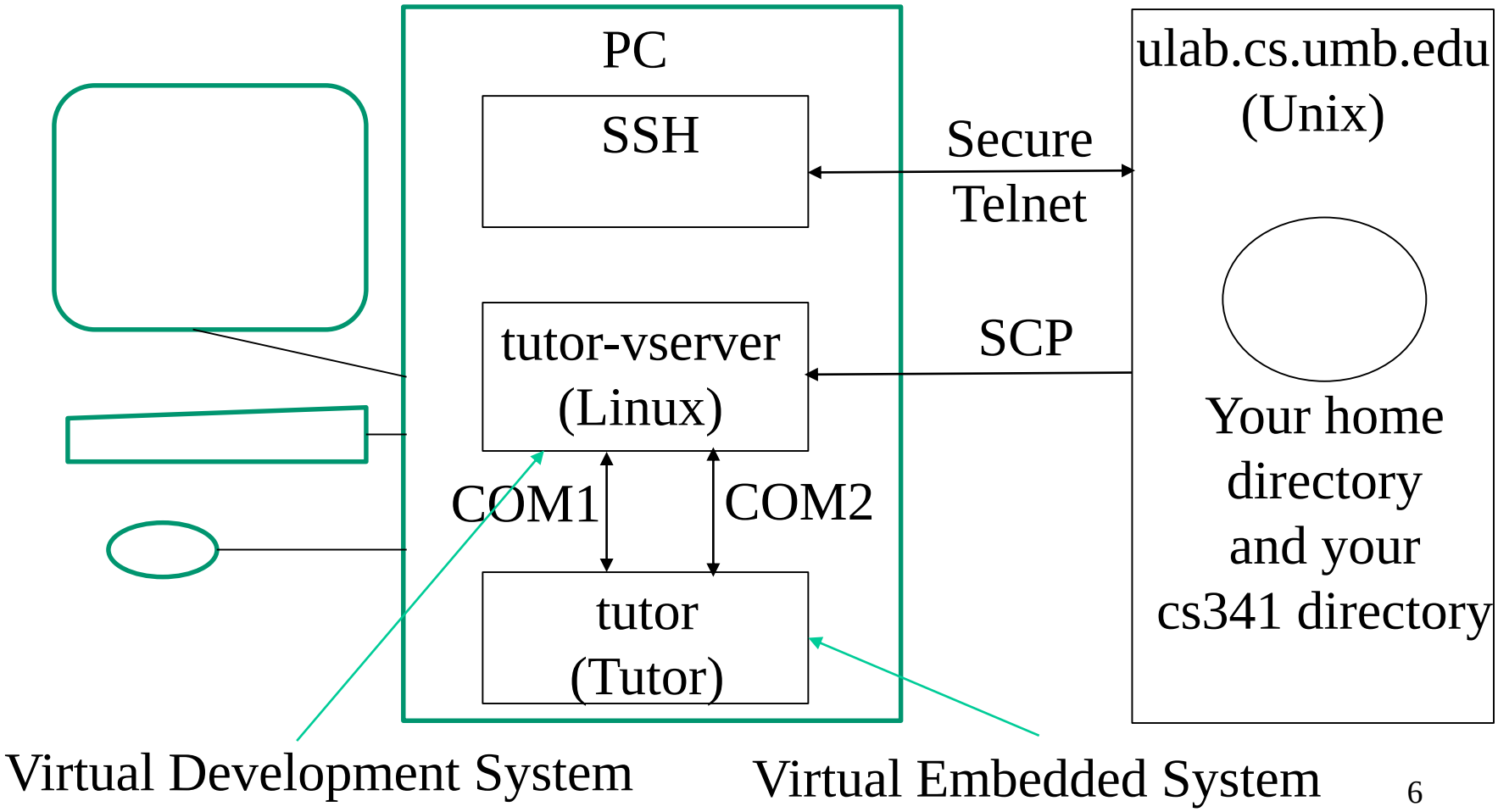




MP Development Environment

- You will also develop C and assembly code for a virtual embedded system on a PC
- You will use two VMware virtual machines
  - One VM provides a Linux development system
  - One VM emulates an embedded system with Tutor
- Tutor is just a debug monitor - not really an operating system like Linux
- Tutor allows operations prohibited by an O/S

MP Development Environment



MP Development Environment

- On your PC:
  - use SSH to login to users@cs.umb.edu
  - rlogin to ulab
- On ulab:
  - copy/edit source files
  - compile and/or assemble them
  - create tutor executable files (.lnx)
- On your PC:
  - openVMWare
  - start tutor-vserver and tutor

MP Development Environment

- On tutor-vserver:
  - use scp to transfer .lnx files to local Linux
  - use mtip to control and monitor the tutor VM

```
(Linux prompt>) mtip –f  filename.lnx
. . .
Tutor>
```

- To download the executable “filename.lnx”

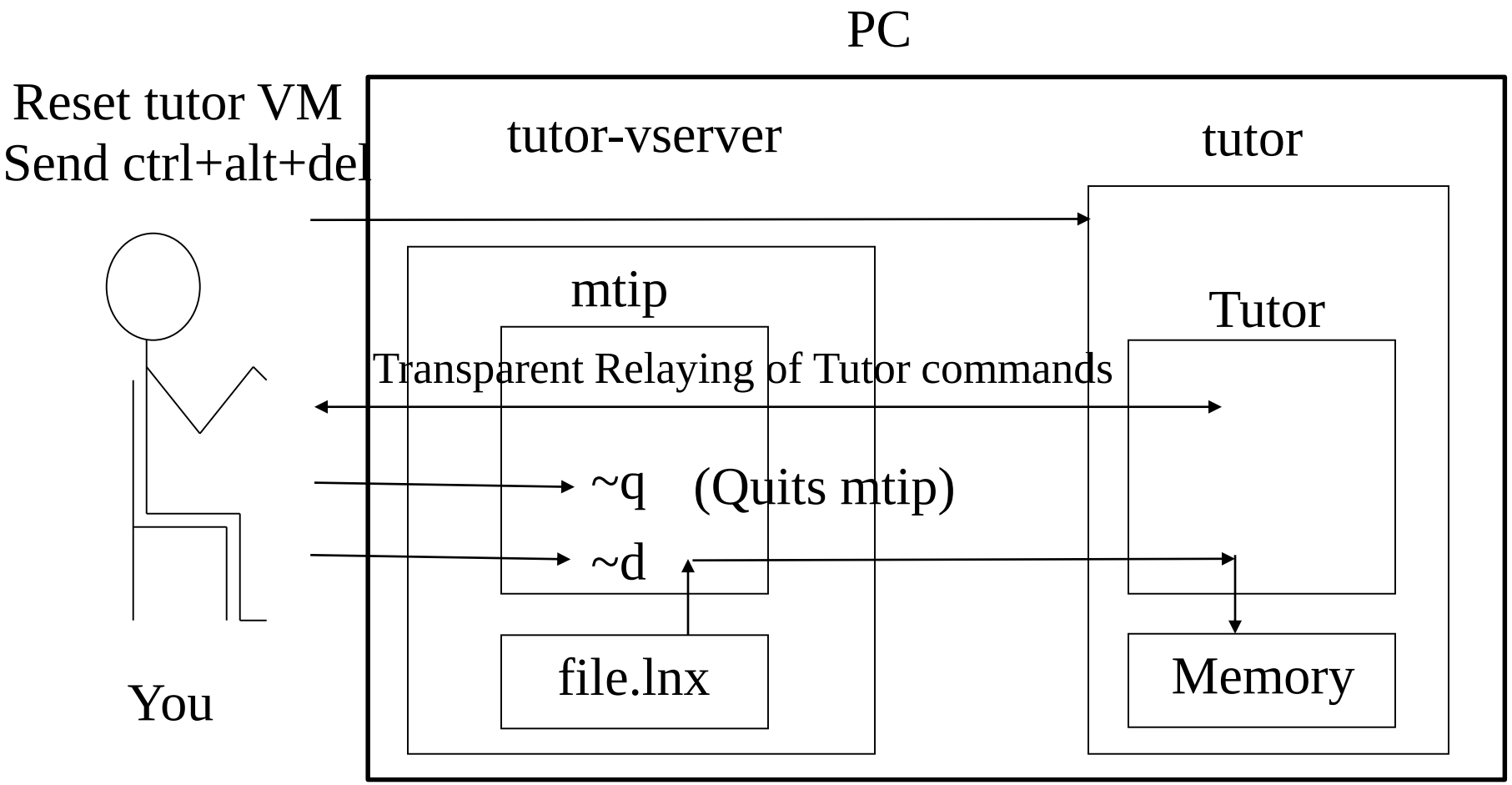
```
Tutor> ~d
```

(Wait for “….Done.” Response)

- To start running the downloaded program:

```
Tutor> go 100100
```

MP Development Environment

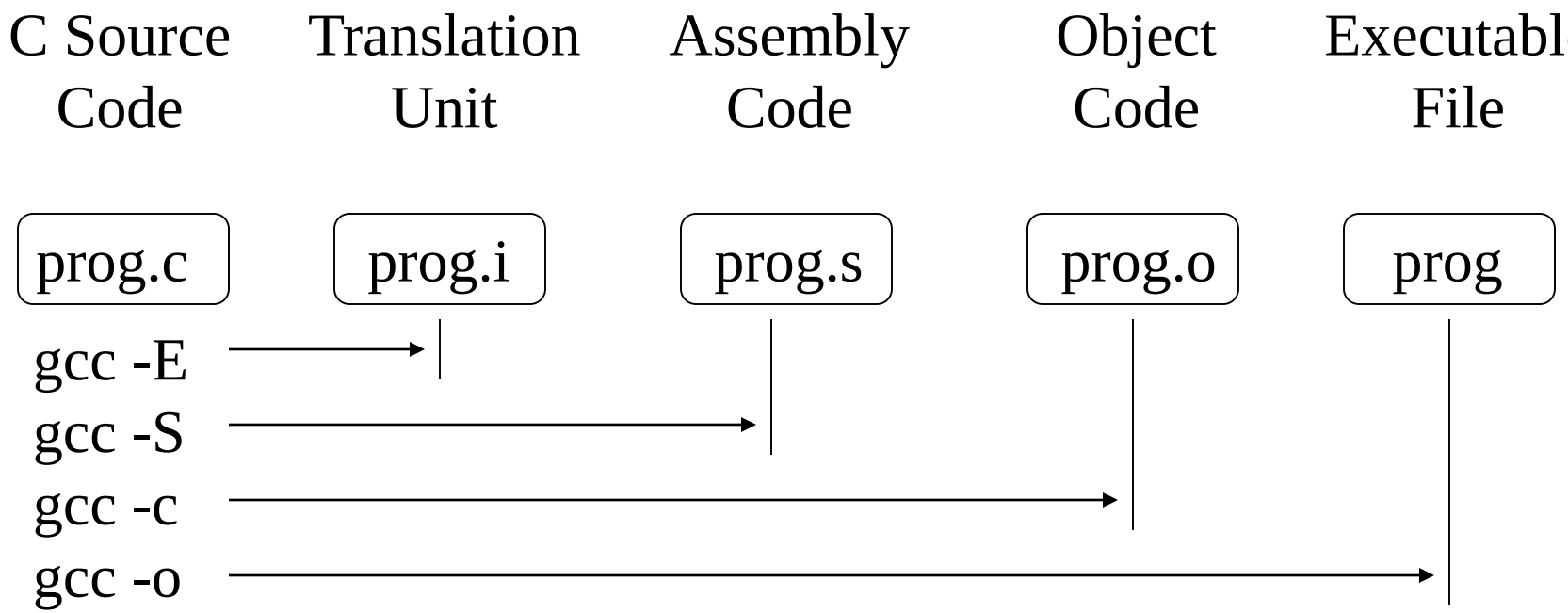


System Build Processes

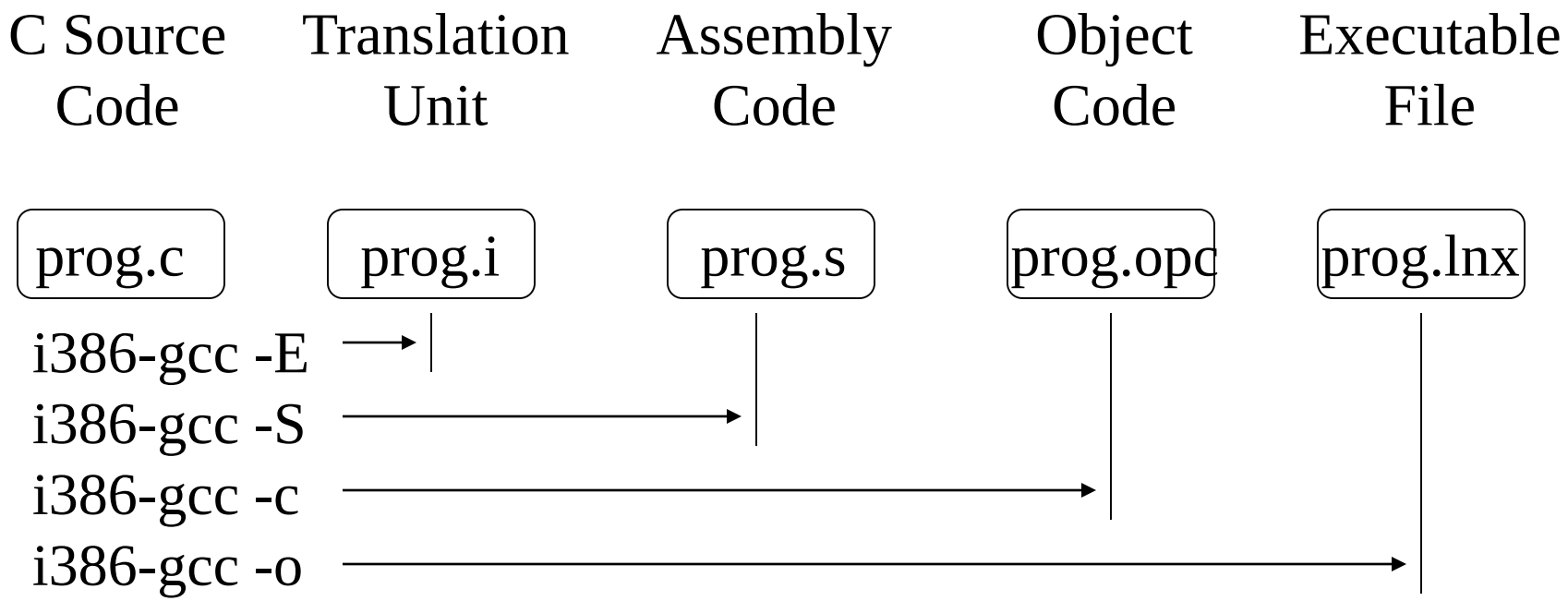
- The system build processes for C and assembly source courses are driven by our makefiles
- However, it is important to understand how the component build steps work with the SW tools:

Unix/Linux	Cross Development for Tutor
gcc	i386-gcc
as	i386-as (informally called “gas”)
ld	i386-ld
nm	i386-nm
objdump	i386-objdump (alias “disas”)

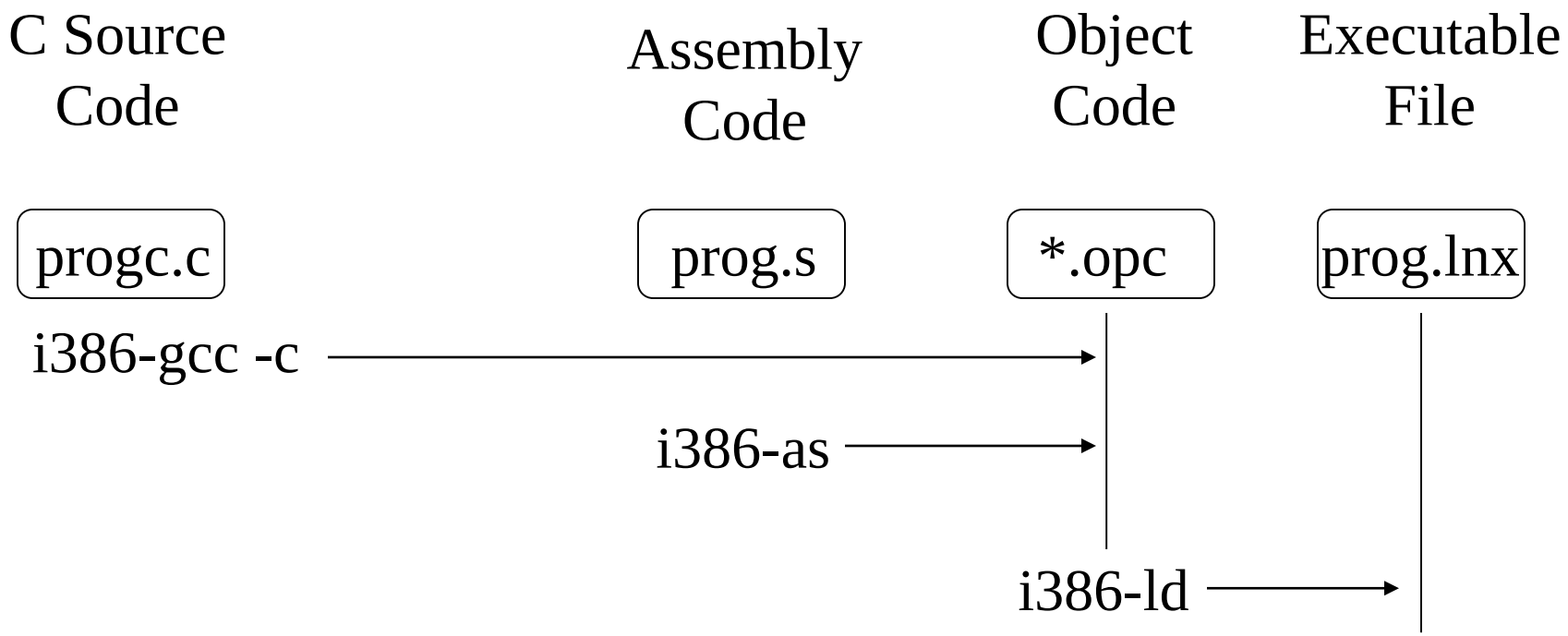
Build for gcc Compilation



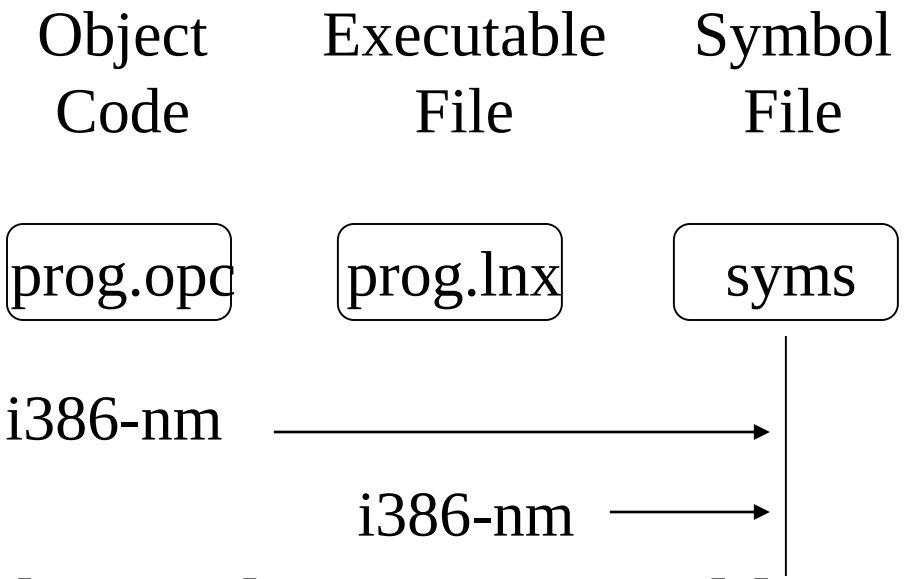
Build for i386-gcc Compilation



Build for i386-gcc and i386-as



Build - Symbol Table Generation



- The syms file shows the memory address assigned for each variable or source label

Example: Test Program

- Make a subdirectory “test” on your cs341
- Copy to that directory the files from: ~bobw/cs341/examples/lecture02
- Compile and run a program named test.c on both ulab/tutor-vserver and Tutor VM

Build/Run test program on Unix

- Create the Linux executable “test”

  ulab(60)% gcc –o test test.c (as in CS240)
- Execute it (avoiding conflict with Unix “test”)

  ulab(61)% ./test
- Follow the program’s directions. It should finish up quickly and hand control back to Linux. You will see a Linux prompt again.

Build/Run test program on Tutor

- Make the Tutor executable, test.lnx .

  ulab(65)% make C=test test.lnx

  – The suffix ".lnx" is a Linux-defined transfer format
- Execute mtip with the executable file you are planning to download and execute

  ```
  ulab(66)% mtip -f test.lnx
  ```
- Hit enter to get the Tutor prompt.
- It's safest to reboot the tutor VM (If a tutor VM ever starts working weirdly, you should reboot it.)

Build/Run test program on Tutor

- Type “~d” to download test.lnx
- To execute the program, when you see the Tutor prompt again, type

```
Tutor> go 100100
```

- Follow the program’s directions. It should finish up quickly and hand control back to Tutor. You will see Tutor prompt again.
- You can run it again by command “go 100100”.
- Type “~q” to quit out of mtip

Analysis of Example

- For Linux or Tutor, the basic process to develop programs is the same - only different in the details
- “Cross-compilation” is defined as the compilation of a program on one computer (UNIX or Linux development host) for execution on another computer (Tutor target machine)
  - We use gcc to generate an executable file that will run on the ulab UNIX system or Linux VM system
  - We use i386-gcc to generate an executable file that can NOT run on Linux but will run on the Tutor VM

Analysis of Example

- Portability
  - Defined as ability to write source code so that it can run on two or more types of machines
  - Requires use of a different compiler/loader and library for each type of machine
- Notice that the library functions called by the test program worked slightly differently on the two types of machines (as noted in the text produced by the test program)

Analysis of Example

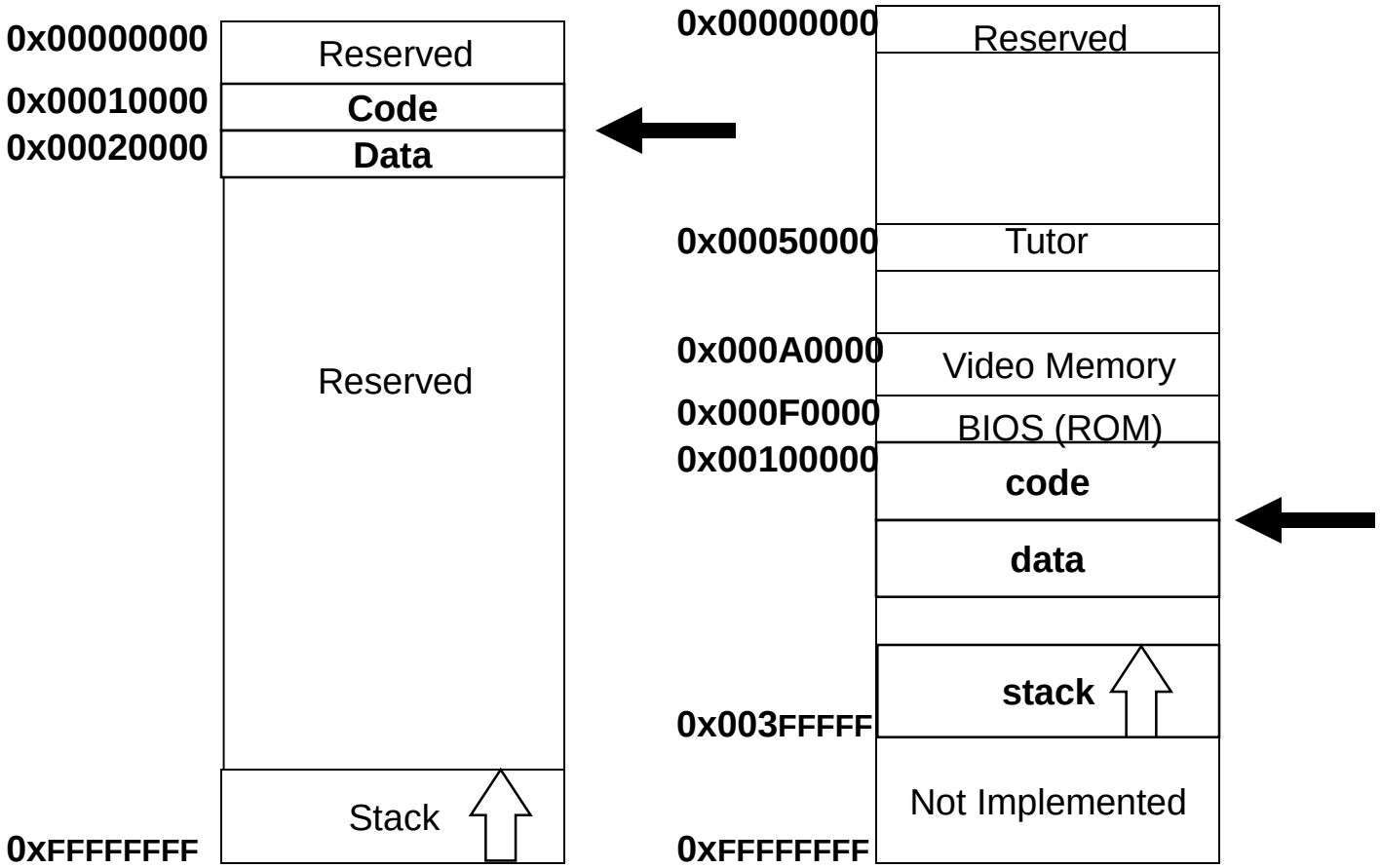
- A key difference between UNIX or Linux VM and the Tutor VM is the presence / absence of an operating system
  - UNIX and Linux are operating systems that run programs in a protected environment. Our code can not do some things such as access hardware directly
  - Tutor is a debug monitor only and does not run programs in a protected environment. Our code can access hardware directly as you'll see later

Machine Project 1

- In mp1, you will add commands to a program that is a simple version of the Tutor program that we just used
- The program is “portable” so that executable runs on ulab UNIX, vserver Linux, and Tutor depending on the build process used
- You will learn about some things about the differences between the UNIX, Linux, and Tutor environments

Two Different Environments

- How is our program loaded into memory?



Sample Run of PC-Tutor on vserver

```
tuser@tutor-vserver:~/cs341/mp1/soln$ pwd
/home/tuser/cs341/mp1/soln
tuser@tutor-vserver:~/cs341/mp1/soln$ ls
tutor.lnx
tuser@tutor-vserver:~/cs341/mp1/soln$ mtip -f tutor.lnx
For command help, type ~?
For help on args, rerun without args
Code starts at 0x100100
Using board # 1
~downloading tutor.lnx

........Done.

Download done, setting EIP to 100100.
Tutor> go 100100
     cmd    help message
      md    Memory display: MD <addr>
      ms    Memory set: MS <addr> <value>
       h    Help: H <command>
       s    Stop
PC-tutor> md 100100
00100100    bc f0 ff 3f 00 bd 00 00 00 00 e8 01 00 00 00 cc ...?............
PC-tutor>
 No such command
PC-tutor> _
```