Discussion of Assignment 9

CSE 2312
Computer Organization and Assembly Language Programming
Vassilis Athitsos
University of Texas at Arlington

Returning Versus Printing

- Regarding the programming tasks in assignment 9, many students have asked: "do we print/return decimal, ASCII, or hexadecimal numbers"?
- If you have this question, it means that you are making a **fundamental** mistake: you are confusing "returning a value" with "printing a value".
- There are multiple ways to answer/clarify this question.
- The simplest one, is to answer with another question: what do the given C functions do?
  - You are asked to implement specific C functions, so you have questions about what your code should be doing, just try to do exactly what the C functions do.

Returning Versus Printing

- Regarding the programming tasks in assignment 9, many students have asked: "do we print/return decimal, ASCII, or hexadecimal numbers"?
- What do the given C functions do?
  - The given C functions do not do any printing.
  - Thus, your functions should not do any printing.
- It is understandable (even recommended) that you may put some code to print stuff for debugging purposes.
  - I would actually recommend that you copy and paste the print_digit and print_number functions to each of your programs, so that you can call print_number for debugging.
- However, once your code is done, you should clean it up and remove, before you submit, any code that does printing.

Returning Versus Printing

- In general, you are asked to implement functions that compute and return something.
- Once you have computed this something, you should store it on register r0.
  - This is the convention we follow for "returning a value".
- ASCII codes are only used for printing.
- When you return a number, the ASCII code of that number is irrelevant.
- It is worth repeating, returning a value has nothing to do with printing a value.

Reading Assembly Code

- Assembly code is painful to read and understand.
- However, you are expected to read any assembly code that you are given.
- How to read assembly code?
  - Start at the beginning.
  - Start mentally executing instructions, one by one.
  - On a piece of paper, write the values of registers and memory addresses that you're using.
  - For each instruction that you "execute" in your mind, update those values on your piece of paper.

Reading Assembly Code

- If you ask me a question of the sort "I do not understand how this piece of code works", I will always ask you to show me how you manually execute this code line by line.
- Not understanding the code means that there is one specific line such that:
  - You do not understand that line.
  - You understand everything before that.
- If you ask me questions where you identify that line, I will be happy to tell you what that line does.
- If you ask me questions of the sort "what does this code do?" I will simply ask you to show me how you manually execute the code.
  - Most of the times, by the time you are done with this exercise, you have answered your own questions.

Existing Assembly Examples

- Look at assembly examples that are available on the slides and the course website.
- A lot of questions can be answered by just looking at those examples.
- For example, consider the factorial function.
  - How does it handle "returning" a value?
  - How does it handle recursive calls?
- Identifying available code that does things similar to what you need to do can save you a lot of time.

The GDB Debugger

- Using the debugger is also a great tool for:
- Understanding code.
- Debugging your own code.
- There are instructions on the course website on how to use the debugger.