Assembly Language Programming with Ubuntu (Jan 2020)

Reviewed in the eighth week of the semester for CPSC 240 (October 2020).

Chapter 1 Introduction

Yes, you should know all of this chapter. It is very short.

Chapter 2 Architecture

Yes, know this chapter except for the following

Section 3.3.1.5 Flags	barely mentioned in class
Section 3.3.1.6 Float registers	a topic for after the midterm

Chapter 3 Data Representation

Yes, know this chapter. In lecture we studied these topics in various levels of detail. For some topics of this chapter we did not go as deep as the author treats the subject. As you read the chapter know that the author goes into more depth than we did in class.

Chapter 4 Program Format

We did most of this chapter. Section 4.6 is a lot different than we did in class. The author shows the name of the function as "_start". The use of this name is mandatory when there is no driver present.

Think of C++. There is always one function whose name is "main". All the other functions have their own names. If there is no driver in another language then the assembly function that begins execution must be called _start.

In our class we did section 4.6 is a different way.

Chapter 5 Tools

This chapter is about the assembler, the linker, the editor, and the debugger. They are all tools to be used during program development. We have not used the open source debugger yet, but you should know about the other tools. You should know the standard bash commands for compiling (C++ and C), assembling (asm), linking, and loading. Loading is a term used for executing. What is the latest standard for C++? The latest standard for C?

Now I know why some of you declared a pointer call 'var'. I thought that was an odd name. In C++ do you name a variable variable? Like this: "int variable = 17;" It just look weird. The author names a pointer to a single value as 'var". It looks odd. However, this author is trying to teach about pointers, and therefore, the name of his pointer is unimportant. He picked 'var' as a convenience.

In summary, the C++ culture says to give descriptive names:

```
int  page_counter;
float  car_price
string  customer_name;
```

The asm culture say to keep the first 13 integers in registers with meaningless names:

```
rcx     ;counter of pages
r8      ;price of car in whole dollars
r13     ;room size square feet
```

In asm culture you have 13 integer variables and you keep reusing those 13. If you actually have 13 integers in active use and you need one more integer then there is a problem. Personally, I cannot remember a program in any language where I was using 13 integers – that's not an array of integers. I am talking about 13 individual integers each with a distinct use. I have never heard of it in my life.

What about volatile registers that unexpectedly. For one point that unexpected change only happens when you call an external function. The author Jorgensen has made a list of the so-called volatile registers in chapter 12 of this book. If you're not calling functions there is nothing to worry about. Quiz question: "I have data in a volatile register and all the stable registers are in use. I need to call an external function. What should I do?".

Omit section 5.2.3

Know section 5.3 Linker. Be aware: the author is using the linker that comes with the nasm package.

Omit section 5.3.3

Omit section 5.4. We make scripts that are far less complex than the ones in this book.

Include section 5.5

Omit section 5.6

Chapter 6 Debugging

Omit all of chapter 6. We will learn it after the midterm.

Chapter 7 The set of instructions in all of X86 assembly

This is all about the instructions we have learned: add, mul, sub, mov, shl, shr, cdqe, cmp, cqo, and, or, not, xor, etc.…

Warning: For most of this chapter it is harder to learn the notation employed here than actually knowing what the operator does. The author describes each instruction in a sophisticated style of describing assembly.

Try reading this chapter. If you can fight through all the weeds you may find some gold nuggets in this chapter.

Look at	7.7.3	jumps
	7.7.4	iteration (loops)

Do you have a list of all the assembly instructions we have use this semester up to the time of the midterm?

Chapter 8 Addressing modes

This chapter explains what kind of operands can each instruction have. You should know this chapter in a basic way, but not in depth.

There are three modes (categories) of operands

- registers
- immediate
- memory

Do you know the meaning of each mode? Do any instructions have no operands?

You might find it interesting to see an entire program without the use of a driver and without access to C++ library functions such as the usual scanf, printf, bubblesort, etc. Such a program is in section 8.3. It just looks odd and feels complex. It is not wrong. It is just different.

Chapter 9 Process stack

He calls it "process stack" and in class I called it the "system stack". Regardless of the name it is a single stack.   You should know this chapter because you have been using the concepts related to a single stack since this course started.   How do you create 800 bytes of available space on top of the system stack without resorting to 100 "push qword 0" instructions? In the stack are numbers stored in big or little endian? In registers are numbers stored in big or little endian? In the stack are strings stored in big or little endian?

Chapter 10 Program Development

This chapter covers the basic steps of building a program: problem understanding, design, detailed design, code, testing, iteration, debugging, and so.   These are important subjects, but they have to be omitted from this course to make room for other topics this semester. Chapter 10 is not on the test.

Chapter 11 Macros

There is not enough time in the semester. Sorry.

Chapter 12 Functions

This is important. This chapter covers subjects that was discussed in lectures many times. How to create a function, how to link it with other object file, how to pass data values into the function and out of the function. In class we called it CCC, but the author has another name for the same principle.

In section 12.8.1 he describes the 6 registers used for passing integers into a called function. This sound just like a lecture in 240 class.

One day in class I talked about how some registers changed values when printf was called and how other register remained fixed in value. There in 12.8.2 the author has all the details about which registers may change their values with another function is called.

In 12.8 there is an explanation of call frames. This subject will become more important as this course progresses. Right now know the basics such as what are the alternate names for a call frame. What marks the boundaries of a call frame? Who can use a call frame? Why do we have such an object? Does C++ use call frames?

From 12.9 to the end of the chapter there is a lot of technical detail not covered in lecture, and therefore, will not be on the midterm test.

Chapter 13 System Services

The kernel of the operating system has functions (routines) can be called application level software. Those functions in the kernel are called system services. This chapter is dedicated to programming in pure assembly. That means there is no driver in C++, and there are no calls to library functions such as scanf and printf. In this course we have skipped “pure assembly” in favor of “hybrid assembly”. This chapter teaches the former style.

So far this semester we are omitting this chapter.

Chapter 14 Multiple source files

We have been making programs where the source code is found in many different files. We have been practicing multiple source files since the beginning of this semester. This chapter is candidate material for the midterm. But I think you already know the subject of this chapter.

Chapter 15 Stack Buffer Overflow

This chapter explains how malicious hacker are able to insert dangerous code into the victim's computer through a technique referred to as "stack buffer overflow". If you are really good at assembly programming and you understand how memory is structured then you can (possibly) take over remote machines. Here you will become acquainted with the stack smashing technique of penetrating remote computers.

This chapter is just fine for those specializing in computer security. For the rest of us the chapter is just leisure reading. Chapter 15 is not on the midterm.

Chapter 16 Command line arguments

This topic is important to know. We all know how to pass parameters from one function to the called function. This chapter explains how to pass arguments from the human user directly into the main function. Unfortunately, there is not enough time to do this chapter. It is not on the midterm.

Chapter 17 Input/Output buffering

Input and output buffers are provided automatically to C++ and C programmers. But what about an assembly programmer creating a strict pure assembly program. How can the pure assembly programmer build in I/O buffers? This chapter explains it all. But we have no time. This chapter will not be on the midterm.

Chapter 18 Floating point operations

Floating point numbers have their own registers and their own instructions. This chapter explains how to use both: registers and instructions. This topic will be studied after the midterm.

Chapter 19 Parallel processing

This is an advanced topic that belongs in a second semester course of assembly programming. How do you create two threads in assembly that will execute in parallel and thereby facilitate your computer to produce more. It is not on the midterm.

Chapter 20 Interrupts

Suppose you are running your own program. If you press control+C you execute a system routine that take precidence over your application program. That is an example of a key-board generated interrupt. This is a very useful chapter describing valuable techniques. I wish there were time to study this chapter this semester. But there is not enough time. This chapter is not on the midterm.

Chapter 21 The Ascii table.

This is for reference. Maybe you want to look up something. You never know. Arn't you glad we have open source tests? Open source testing means you may use anything at your disposal.

Chapter 22 The X86 set of instructions

The author says that this is a list of all the assembly instructions used in this book. That is nice. Maybe you will be able to find the instruction that you need in your homework assignment.

The author shows lots of instructions, but I think he does not have them all. The total number of instructions is 709, and in the 240 class we learn to use about 30 of them. I haven't counted how many the author includes in Chapter 22.

Chapter 23 System Services

This is a companion to Chapter 13. If you want to know pure assembly programming then you will need both Chapters 13 and 23.

Chapter 24 Answers to quiz questions

At the end of each chapter of this book there are practice quiz questions. That helps the readers measure his or her understanding of the chapter. All the answer to all the questions are here in Chapter 24.