



Assembly Language – Introduction

- Learning Objectives
 - Explain what assembly language is
 - Define
 - Registers
 - Instruction
 - Operands
 - Produce assembly from C



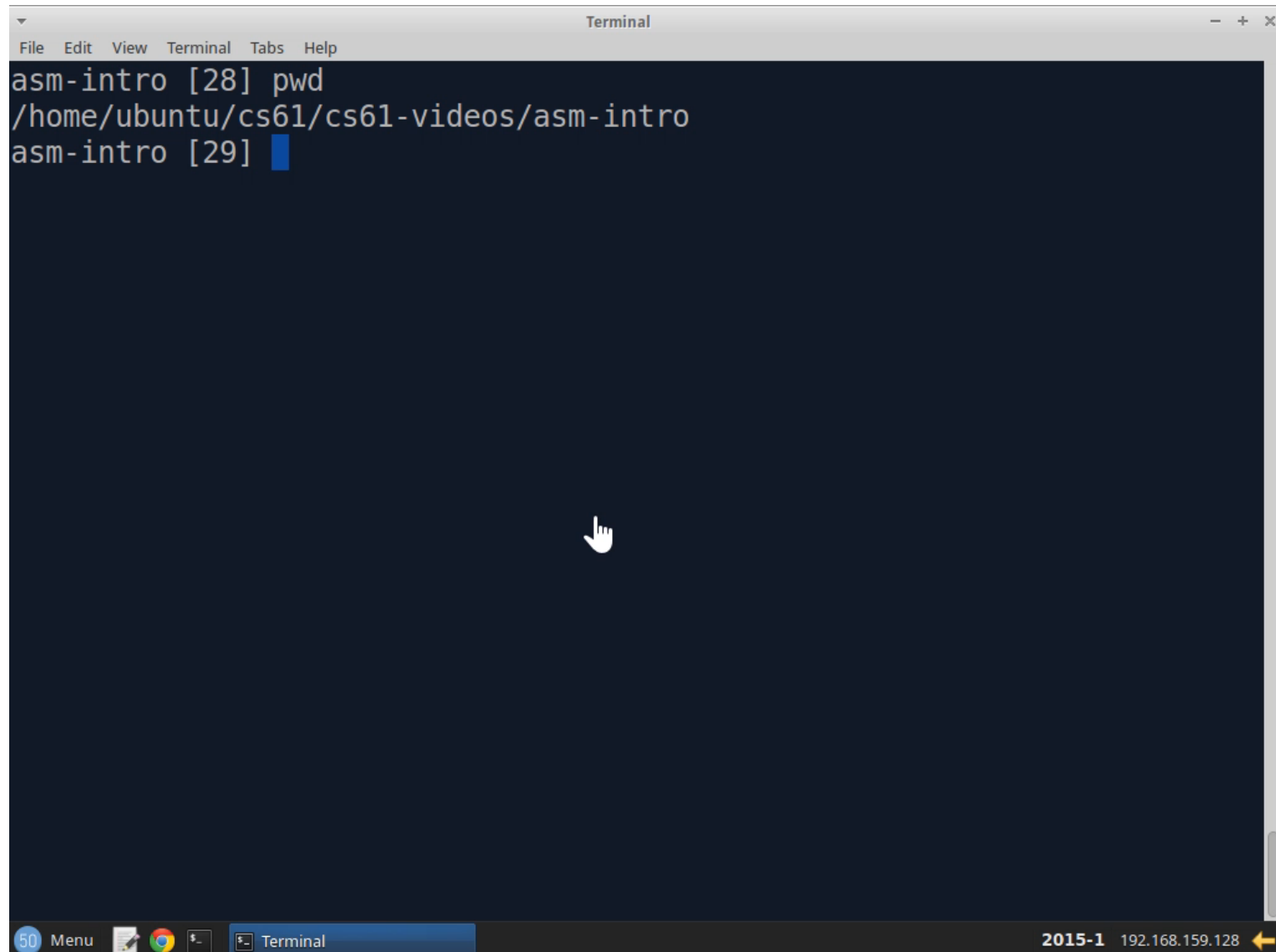
What is assembly?

- Yet another layer of abstraction!
- When you strip away C, the assembly language is a human readable representation that more closely matches the hardware.
- Typically each assembly instruction corresponds to a machine instruction.
- Assembly doesn't really manipulate variables; it expresses computation in terms of:
 - Registers
 - Memory
 - Instructions



A Note on our Assembly

- We are using Intel x86 assembly and assuming a 32-bit processor.
- The 3rd edition of the book uses a 64-bit architecture in most sections, but still has some remnants of the 32-bit architecture as well.
- The two are quite similar, but keep in mind that we're using 32 bits.
- Three ways to see assembly output:
 - `cc -S -O3 x.c`
 - `objdump -d x.o`
 - In gdb: `disassemble <address>`



A terminal window titled "Terminal" with a menu bar containing "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal content shows the command "pwd" being executed in the directory "asm-intro [28]", resulting in the output "/home/ubuntu/cs61/cs61-videos/asm-intro". The prompt then changes to "asm-intro [29]" with a blue cursor. The terminal is part of a desktop environment with a taskbar at the bottom showing a clock at 6:00, a "Menu" button, and icons for a terminal, Chrome, and a shell. The system status bar at the bottom right displays "2015-1" and the IP address "192.168.159.128" with a yellow arrow pointing left.

```
asm-intro [28] pwd
/home/ubuntu/cs61/cs61-videos/asm-intro
asm-intro [29] █
```



Example

```
.file    "1.c"
.text
.globl   func
.align   16, 0x90 ←
.type    func,@function

func:    # @func
# BB#0:
    ret

.Ltmp0:
.size    func, .Ltmp0-func

.ident   "Ubuntu clang version 3.4-1ubuntu3 ..."
.section ".note.GNU-stack","",@progbits
```



Registers

- **Registers** are fast memory in the processor.
 - Processors typically execute most instructions in a single cycle; accessing memory can take 10's or 100's of cycles; placing data in registers allows the processor to execute things more quickly.
 - Most processors have a few tens of registers.
 - The Intel x86 has eight 32-bit registers:
 - %eax, %ebx, %ecx, %edx, %edi, %esi, %esp, %ebp
 - There are conventions for how some of the registers are used.
 - For example: %esp is the stack pointer
 - %ax is the lower 16 bits of register %eax
 - %al is the low order byte of register %eax
 - %ah is the second lowest order byte of %eax



Kinds of instructions

- Move data around
- Perform arithmetic operations
- Perform logical operations
- Compare things (sets **condition** flags)
- Flow control