

# CS 161: Introduction to Programming and Problem-solving

**Warren Harrison**

*Solving Problems with  
Computers*

# Processes

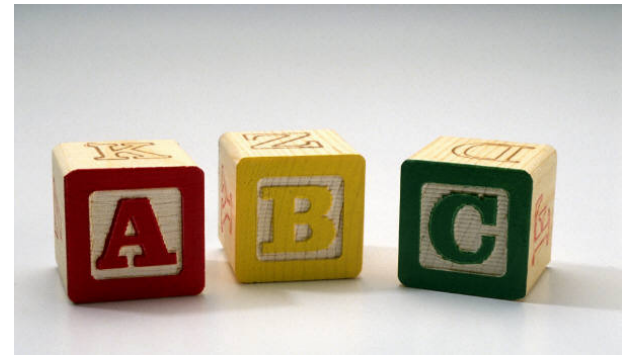
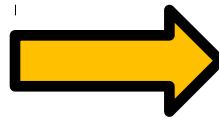
- the way things get done ...
- we can view a process as a recipe for obtaining a result
- often a process converts *inputs* into *outputs* through *processing*

# Inputs, Processing, Outputs (IPO)

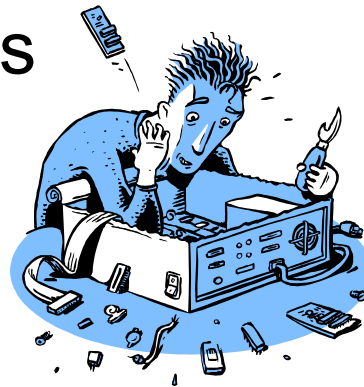
- inputs are things that feed into the processing
- processing involves actions taken to *transform* the inputs
- outputs are what comes out of the processing after the inputs are transformed

# Examples of Physical Inputs and Outputs



- change form




- assemble pieces




# Examples of Virtual Inputs & Outputs

- change form
  - CRN  Course Name and Number
- assemble pieces
  - Hours and Rate  Paycheck

# Processing: Transformation of Inputs to Outputs

- Producing a paycheck
  - input is hours worked and rate of pay
  - output is net pay
  - transformation is:
    - Hours Worked \* Rate of Pay – Deductions  Net Pay

# Processing: Transformation of Inputs to Outputs

- Producing a paycheck
  - input is hours worked and rate of pay
  - output is net pay
  - transformation is:
    - Hours Worked \* Rate of Pay – Deductions  Net Pay
  - where do deductions come from?
  - how about overtime?
  - more complex than we thought at first ...

# A Paycheck

from <http://themint.org/images/paycheck.gif>

123 - John R. Doe				Pay Period 06/02/06 to 06/16/06			Required Deductions		
Earnings				Federal Income Tax	00.00		00.00		
Hours	Rate	This Period	YTD	FICA - Medicare	06.08		12.16		
50	9.00	450.00	900.00	WI State Income Tax	00.00		00.00		
Gross Pay		450.00	900.00	FICA - Social Security	25.92		51.84		
				Other Deductions					
				Health Insurance	00.00		00.00		
				401k	00.00		00.00		
				Parking	00.00		00.00		
				NET PAY			\$418.00	\$836.00	

Your Employer  
1234 Some Street  
Milwaukee, WI ZIPCODE

Check Number: XXXXXX  
Pay Date: 06/19/06

PAY \*\*\*Four hundred eighteen dollars and 00 cents\*\*\*\*\*\$418.00

To the Order of  
John R. Doe  
555 Some Street  
Milwaukee, WI ZIP CODE





# Producing a Paycheck, steps 1 & 2

- multiply *hours worked* by *hourly rate* to get *gross pay*
- if *hours worked* is more than 40, multiply the number of *hours worked* over 40 by half the *hourly rate* and add to *gross pay*

# Producing a Paycheck, step 3

- multiply *gross pay* by 12 to get *annual income* and compute *income tax withholdings* by multiplying *gross pay* by the tax withholding rate, which, if *annual income* is:
  - \$2,200 or less is 0%
  - more than \$2,200 and less than \$11,125 is 10%
  - between \$11,125 and \$38,449 is 15%
  - between \$38,450 and \$90,049 is 25%
  - between \$90,050 and \$185,449 is 28%
  - between \$185,450 and \$400,549 is 33%
  - between \$400,550 and \$402,199 is 35%
  - over \$402,200 is 39.6%

# Producing a Paycheck, steps 4, 5 & 6

- multiply *gross pay* by 6.20% for *FICA* and 1.45% for *Medicare*
- subtract *income tax withholdings*, *FICA* and *Medicare* from *gross pay* to get *net pay*
- write the employee's name on the check as the payee, write the net pay in numeric format in the numeric amount area, write the net pay in written format in the written amount area, date the check and sign it.

# Designing a Process

- what outputs do you want?
- what inputs do you need in order to produce those outputs?
- what processing is required to turn those inputs into the desired outputs?

# The Role of Software in Processes

- activities in a process can be:
  - manual
  - automated
  - combination of manual and automated
- automation works well when tasks are:
  - straight forward and well understood
  - require little judgment
  - don't change frequently

# What is Software?

- instructions to a computer
- mainly involve transferring information within a computer, performing calculations, and making simple decisions
- usually involve obtaining information from the user or a persistent data store; processing that information; and either storing it in a persistent data store or outputting it to a screen or printer

# What is a Computer?

- A computer is a machine (hardware) that can interpret and execute a sequence of instructions
- Computers are known as ***General Purpose Computers*** because they can be used for anything



# What is Software?

- A sequence of instructions is called a program (*software*)
- Software is what allows a computer to be a general purpose machine.
- You need both hardware and software to accomplish anything with a computer



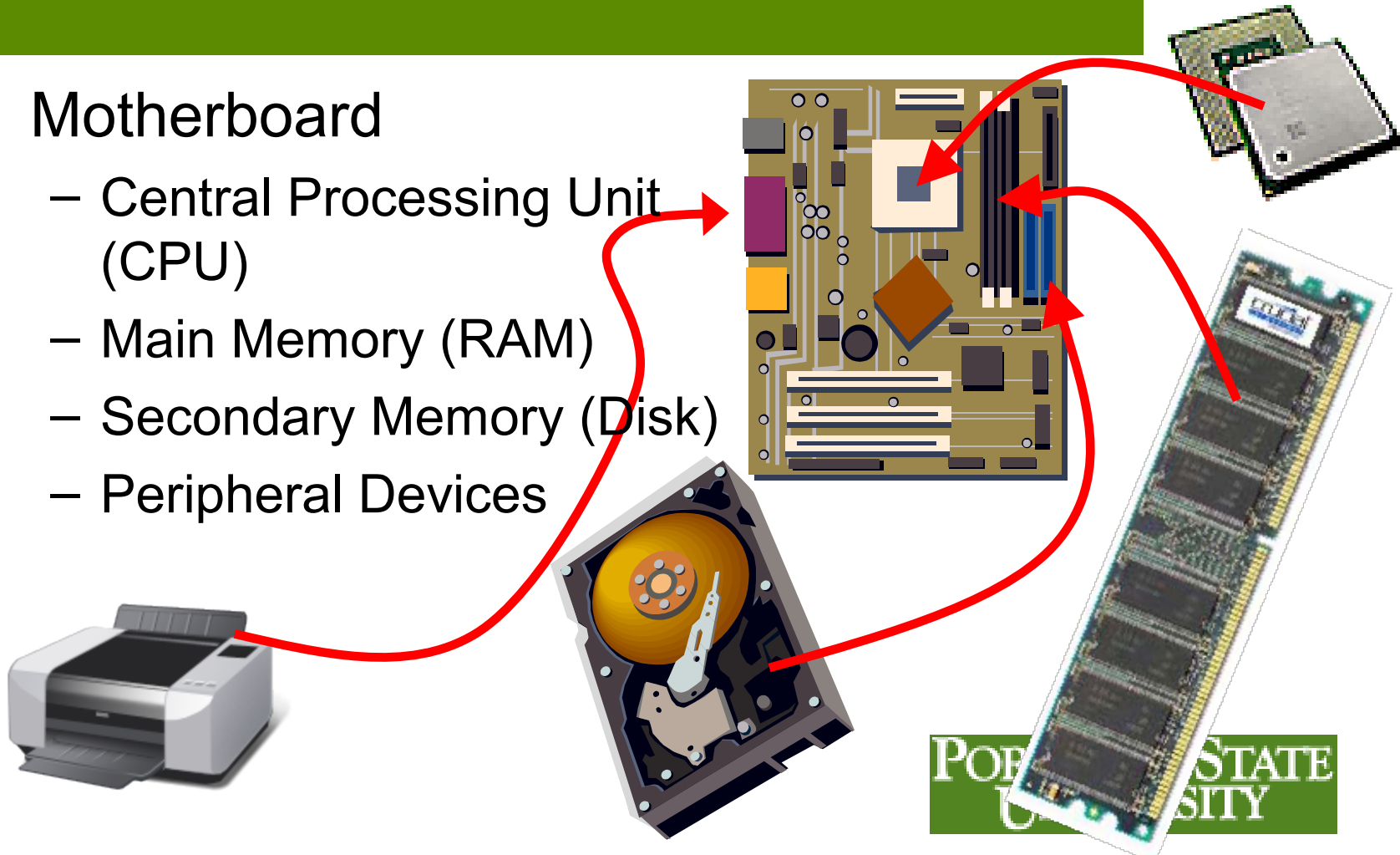
# General Purpose Computer

- By changing the software, the function of a computer changes:
  - word processing
  - spreadsheets
  - numerical analysis
  - games
  - communications



# Computer Organization

- Motherboard
  - Central Processing Unit (CPU)
  - Main Memory (RAM)
  - Secondary Memory (Disk)
  - Peripheral Devices



# The Central Processing Unit (CPU)

- The CPU can only understand a limited number of *simple* instructions - such as "add two numbers" or "multiply two numbers".
- The set of instructions a CPU can understand is known as that computer's **instruction set**.
- The number of instructions the CPU can execute in one second determines the CPU's speed – 1 Megahertz = 1 million instructions/second

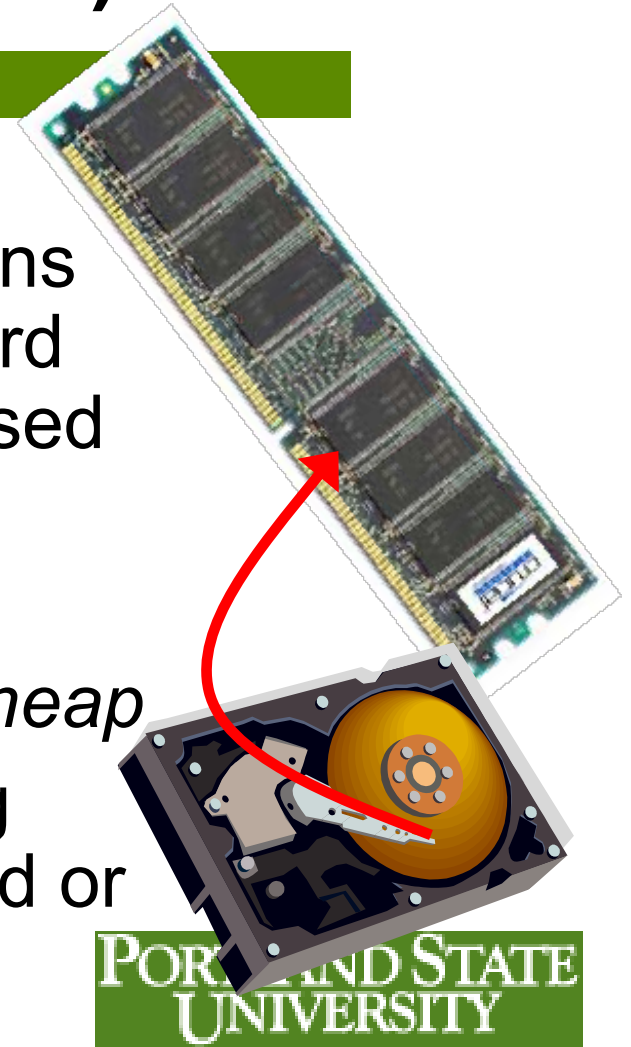
# Main Memory (RAM)

- The instructions are located in RAM while they're waiting to be executed by the CPU
- RAM is *fast*, but (relatively) *expensive*
- RAM is *volatile* (contents disappear when power is removed)



# Secondary Memory (Disk)

- Only one program can be executed at a time - Instructions and data are stored on the hard disk when they aren't being used
- When you run a program, it's *loaded* into RAM
- Disk is (relatively) *slow* and *cheap*
- Disk is *persistent* – everything stays whether power is applied or not



# How the CPU/RAM Works

- Program instructions are loaded into RAM from the disk
- The CPU takes one instruction at a time, in sequence, and executes it
- Instructions are usually represented as a numeric operation code (**Opcode**)

# Computer Instructions

- If 06 means “add two numbers”, we need to also specify what numbers we are supposed to add
- The operation to be performed is known as the **Opcode**, the data on which the instruction is to be performed is called the **Operand(s)**.
- Let’s say we want to add 10 to 100, our instruction might look like this: **06010100**

# Where Does the Result Go?

- **06010100** adds 10 by 100. Where does the answer go?
- The result of the operation goes to the “**Accumulator**”
- This is a special holding location in the CPU



# A Program to Keep Your Checkbook Balance

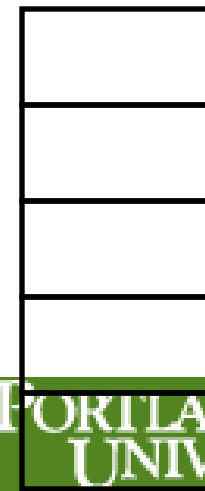
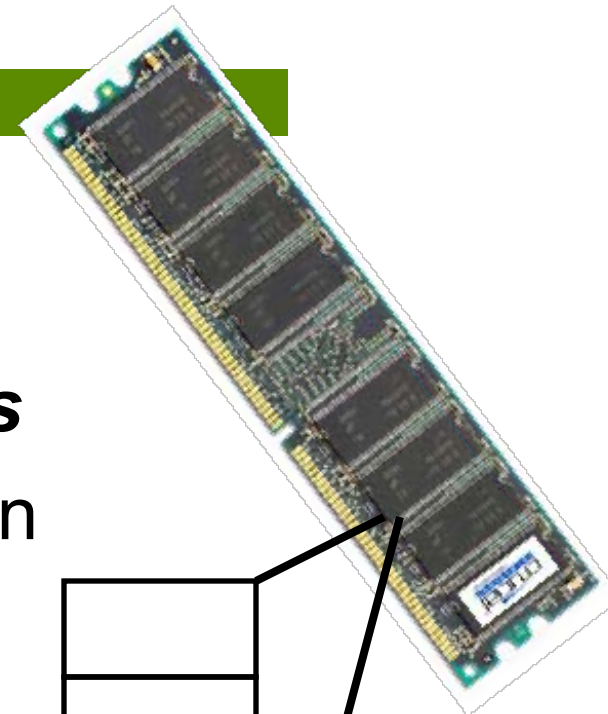
- 06800025
- 06775037

Check Number	Date	Transaction Description	Payment/ Debit (-)	<input checked="" type="checkbox"/>	Deposit/ Credit (+)	Balance
		Balance Forward				27.42
DEP	7/20	Deposit paycheck			800.00	+800.00 827.42
DC	7/24	Dinner	25.00			-25.00 802.42
476	7/26	Utilities	37.42			-37.42 765.00
					00	

Not very useful – in order to write the second instruction you'd need to do the computation specified by the first instruction

# Main Memory

- RAM consists of many “boxes”, that can each hold a number
- Each box has a unique **address**
- Instructions are stored in RAM in the sequence in which they are to be executed
- Number of “boxes” is the **size** of the RAM – 1 Gigabyte has 1,000,000,000 boxes



# Use the *CONTENTS* of the Memory Locations as the Operands

- Instead of an instruction like **06800025** referring to “800” and “25” it could refer to the numbers in locations 800 and 25.
- This means that **06800025** could add many different numbers, depending on what is placed into those memory locations

# Storing Data in Main Memory

- Specifying the address of the memory location in which a number is stored instead of the number itself is known as **indirect referencing**.
- Provides a general solution since you can write an instruction that will perform many different computations depending on what values are in the memory locations referenced in the instruction.

# A Johnniac Simple Machine Simulator (JSMS) Instruction

- **Syntax:**
  - OPCODE [2 digits]
  - OPERAND [3 digits]
- **Semantics:**
  - Perform the operation indicated by the Opcode to the contents of the memory location specified by the Operand and place into the Accumulator

# The Johnniac Simple Machine Simulator (JSMS) Instruction Set

- 00 HALT – Stop Execution
- 01 LOAD – Put c(Operand) into Acc
- 02 STORE – Put c(Acc) into the Operand Loc
- 03 ADD – Add c(Operand) to c(Acc), result in Acc
- 04 MULTIPLY - Multiply c(Operand) by c(Acc), result in Acc
- 05 DIVIDE - Divide c(Operand) by c(Acc), result in Acc
- 06 SUBTRACT - Subtract c(Operand) from c(Acc), result in Acc
- 07 TEST – if c(Acc) is not zero, continue execution at Operand
- 08 GET – take 5 digit number from keyboard and place into Operand
- 09 PUT – display c(Operand) on screen
- 10 NOOP – dummy operation

# A Simple Program

*tells you when you're out of money by displaying a 99999 code...*

LOC	PROGRAM	LOC	RAM (MEMORY)
000	08018 # get	010	99999
001	01018 # load	011	00000
003	08017 # get	012	00000
004	06017 # subtract	013	00000
005	07003 # jump if not zero	014	00000
006	09010 # display code	015	00000
007	00000 # halt	016	00000
008	00000	017	00000
009	00000	018	00000

**ACCUMULATOR: 00000**

# Making Programming Easier

- It's easy to get confused – especially in keeping the opcodes and operand locations straight.
- Programmers assemble small building blocks of low level instructions into individual high level instructions to create the software we use
- Programs have been written to make this easier – these program are called ***compilers*** and the “language” they use is called a ***programming language***



# Many, Many Different Programming Languages

- BASIC
- C
- C#
- C++
- Java
- Ada
- Perl
- Python