Programming

- Why is programming fun?
  - Finally, there is the delight of working in such a tractable medium. The programmer, like the poet, works only slightly re-moved from pure thought-stuff. He builds his castles in the air, from air, creating by exertion of the imagination. Few media of creation are so flexible, so easy to polish and rework, so readily capable of realizing grand conceptual structures.

Source: Frederick P. Brooks, Jr. *The Mythical Man-Month Essays on Software Engineering.*

Announcements

- Undergraduate Research Symposium
  - Friday
  - How many attended?

Announcements

- Project 2B
  - Due on Wednesday before 12 Noon
    - If you don't submit the quiz before 11, your answers are gone!!
    - ***Aim at submitting quiz before 11***

Announcements

- Labs this week
  - Monday-Tuesday
    - Finish up project 2B
  - Wednesday-Thursday
    - Grading spreadsheet that will calculate your current grade in the class

Getting Help

Exercise 4

- JavaScript Exercise 4
  - *Describe how you use a for loop to cycle through radio buttons to find the one that has been checked.*

FIT100

Exercise 4

```
<label for="giraffe">Giraffe</label><br />
<input type="radio" id="giraffe"
  name="animals" />
<label for="zebra">Zebra</label><br />
<input type="radio" id="zebra"
  name="animals" />
<label for="lion">Lion</label><br />
<input type="radio" id="lion"
  name="animals" />
```

FIT100

Exercise 4

```
<label for="giraffe">Giraffe</label><br />
<input type="radio" id="giraffe"
  name="animals" />
<label for="zebra">Zebra</label><br />
<input type="radio" id="zebra"
  name="animals" />
<label for="lion">Lion</label><br />
<input type="radio" id="lion"
  name="animals" />
```

FIT100

Exercise 4

```
<label for="giraffe">Giraffe</label><br />
<input type="radio" id="giraffe"
  name="animals" />
<label for="zebra">Zebra</label><br />
<input type="radio" id="zebra"
  name="animals" />
<label for="lion">Lion</label><br />
<input type="radio" id="lion"
  name="animals" />
```

FIT100

Exercise 4

```
<label for="giraffe">Giraffe</label><br />
<input type="radio" id="giraffe"
  name="animals" />
<label for="zebra">Zebra</label><br />
<input type="radio" id="zebra"
  name="animals" />
<label for="lion">Lion</label><br />
<input type="radio" id="lion"
  name="animals" />
```

FIT100

Exercise 4

```
for (var i = __; i < 3;  i++)
{
   if(____________)
  {
     //coding goes here
  }
}
```

FIT100

Exercise 4

```
for (var i = __; i < 3;  i++)
{
   if(____________)
  {
     //coding goes here
  }
}
```

Exercise 4

```
for (var i = __; i < 3;  i++)
{
   if(            )
   {

   }
}
```

Exercise 4

```
for (var i = __; i < 3;  i++)
{
   if(document.getElementById )
   {
      //do something here
   }
}
```

Exercise 4

```
for (var i = __; i < 3;  i++)
{
   if(document.getElementById.checked )
   {
      //do something here
   }
}
```

Exercise 4

```
for (var i = __; i < 3;  i++)
{
   if(document.getElementById.checked == true)
   {
      //do something here
   }
}
```

Following Instructions

Principles of Computer Operation, or How Computers Work

Instruction Execution Engines

- What computers can do
  - Perform or execute instructions to process information
    - The computer must have instructions to follow

Short list!

9-18

Instruction Execution Engines

- What computers can't do
  - Have no imagination or creativity
  - Have no intuition
  - Have no sense of irony, subtlety, proportion, decorum, or humor
  - Are not vindictive or cruel
  - Are not purposeful
  - Have no free will
  - Recent movies: Terminator, Matrix, AI

Long list!

9-19

Anatomy of a Computer

- Computers have five basic parts or subsystems
  - Memory, control unit, arithmetic/logic unit (ALU), input unit, output unit

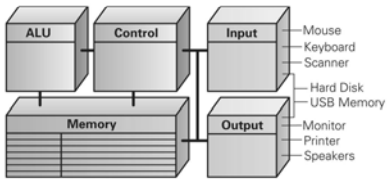

Figure 9.2. The principal subsystems of a computer.

9-20

Memory

- *Memory* stores the program running and the data on which the program operates
- Properties of memory:
  - Discrete locations—1 byte per location!
  - Addresses—For every memory location (byte)
    - whole numbers starting with zero
  - Values—Memory locations store values.
  - Finite capacity—Limited size—data may not "fit" in the memory location.
    - Overflow conditions, buffer overruns

9-21

Byte-Size Memory Location

- A commonly used diagram of computer memory represents the discrete locations as boxes (1 byte each).
- Address of location is displayed above the box.
- Value or contents of location is shown in the box.

0	1	2	3	4	5	6	7	8	9	10	11	
100	T	h	a	N	K	$	*	4	b	d	a	...

Figure 9.3. Diagram of computer memory illustrating its key properties.

9-22

Memory (cont'd)

- 1-byte memory locations can store one ASCII character, or a number less than 256 (0 - 255)
- Programmers use a sequence of memory locations together, ignoring the fact that they all have different addresses
  - Blocks of four bytes are used as a unit so frequently that they are called memory "words"

9-23

Random Access Memory (RAM)

- "Random access" means the computer can refer to (access) the memory locations in any order
- Often measured in
  megabytes (MB) – millions of bytes or
  gigabytes (GB) – billions of bytes
- Large memory is preferable because there is more space for programs and data (which usually equates to less I/O)

9-24

Control Unit

- Its circuitry fetches an instruction from memory, decodes the instruction, and fetches the operands used in it
  - A typical instruction might have the form

    ADD 4000, 2000, 2080 **op dest, src1, src2**
  - This instruction asks that the numbers stored in locations 2000 and 2080 be added together, and the result stored in location 4000 **[4000] = [2000] + [2080]**
  - Data/Operand Fetch step must get these two values and after they are added, Result Return/Store step will store the answer in location 4000

9-25

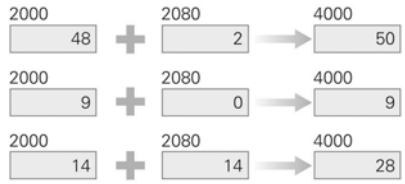

Figure 9.4. Illustration of a single ADD instruction producing different results depending on the contents of the memory locations referenced in the instruction.

9-26

Arithmetic/Logic Unit (ALU)

- Performs the math
  - A circuit in the ALU can add two numbers
  - Other circuits do multiplication, comparisons, etc.
- Instructions that just transfer data usually don't use the ALU
- Data/Operand Fetch step of the Cycle gets the values that the ALU needs to work on (operands)
- After the ALU completes an operation, the answer is moved from the ALU to the destination memory address specified in the instruction

9-27 * taxDue = taxRate[WA] * subtotal;

Input Unit and Output Unit (I/O)

- The wires and circuits through which information moves into and out of a computer
- *Peripherals*
  - Connect to the computer input/output ports.
  - Not considered part of the computer, but specialized gadgets that encode or decode information between the computer and the physical world.
    - Modems, monitors, scanners, printers, keyboard, mouse, digitizing pad, mic, speakers

9-28

The Peripherals

- Keyboard encodes keystrokes we type into binary form for the computer
- Monitor decodes information from the computer's memory and displays it on a lighted, colored screen
- Disks drives are used for both input and output—storage devices where the computer puts away information when it is not needed, and can retrieve from when it is needed again

9-29

A Device Driver for Every Peripheral

- "Dumb" devices provide basic physical translation to or from binary signals.
- Additional information from the computer is needed to make it operate intelligently.
- e.g., computer receives information that user typed shift and w at the same time. It converts to a capital W. The software that converts is called the device driver.

9-30

The Program Counter: The Pc's PC

- How does the computer determine which step to execute next?
- Address of the next instruction is stored in the Control Unit in the *program counter (PC).*
- Because instructions use 4 bytes of memory, the next instruction must be at PC + 4, 4 bytes further along in the sequence (in general).
- Computer adds four to the PC, so when the F/E Cycle gets back to Instruction Fetch step, the PC is "pointing at" the next instruction.

9-31

Branch and Jump Instructions

- The instruction may include an address to go to next. This changes the PC, so instead of going to PC +4 automatically, the computer "jumps" or "branches" to the specified location.

9-32

Instruction Interpretation

- Process of executing a program
  - Computer is interpreting our commands, but in its own language
- Before the F/E Cycle begins, some of the memory locations and the PC are visible in the control unit

9-33

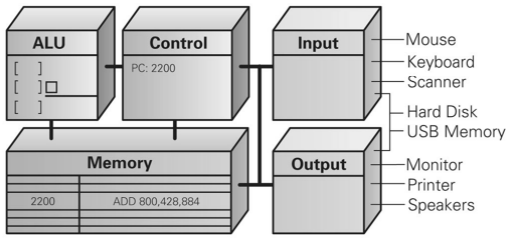

Figure 9.5. *Computer before executing an ADD instruction.*

The Fetch/Execute Cycle

- A five-step cycle:
  1. Instruction Fetch (IF)
  2. Instruction Decode (ID)
  3. Data Fetch (DF) / Operand Fetch (OF)
  4. Instruction Execution (EX)
  5. Result Return (RR) / Store (ST)

9-35

Animation

- Fetch/Execute Cycle

Cycling the F/E Cycle

- Computers get their impressive capabilities by executing many of these simple instructions per second
- The Computer Clock: Determines rate of F/E Cycle
  - Measured in gigahertz (GHz), or billions of cycles per second

9-37

How Important is Clock Speed?

- Modern computers try to start an instruction on each clock tick
- Pass off finishing instruction to other circuitry (*pipelining*)
  - Five instructions can be in process at the same time
- Does a 1 GHz clock really execute a billion instructions per second?
  - Not a precise measurement. Computer may not be able to start an instruction on each tick, but may sometimes be able to start more than one instruction at a time

9-38