Chapter 9: Following Instructions: Principles of Computer Operation

Fluency with Information Technology
Third Edition

by
Lawrence Snyder

Instruction Execution Engines

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

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)

Anatomy of a Computer

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

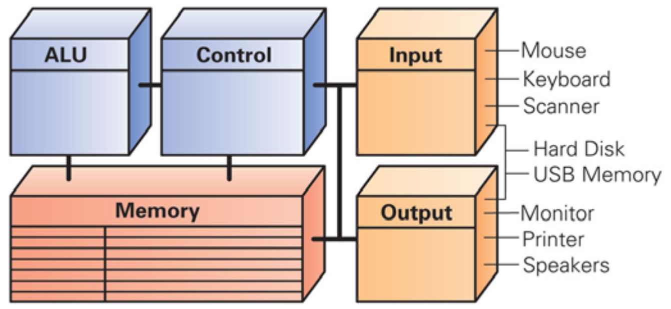


Figure 9.2. The principal subsystems of a computer.

Memory

- *Memory* stores the program running and the data on which the program operates
- Properties of memory:
  - Discrete locations. Each location consists of 1 byte.
  - Addresses. Every memory location (byte) has an address (whole numbers starting with zero).
  - Values. Memory locations record or store values.
  - Finite capacity. Limited size—programmers must remember that the data may not "fit" in the memory location.

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.

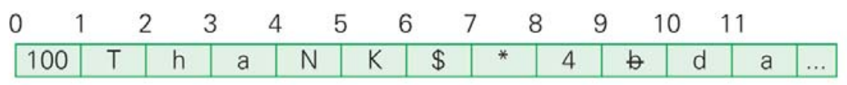


Figure 9.3. Diagram of computer memory illustrating its key properties.

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"

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)

Control Unit

- Hardware implementation of the Fetch/Execute Cycle
- 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

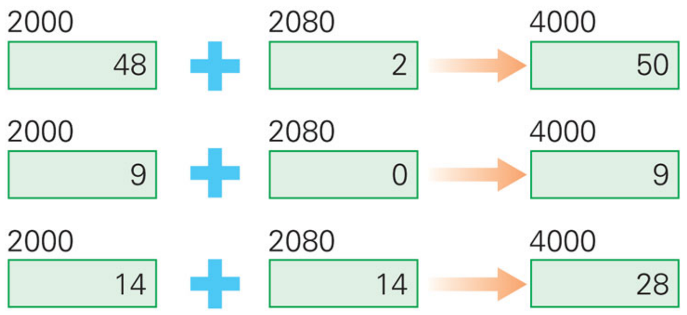


Figure 9.4. *Illustration of a single ADD instruction producing different results depending on the contents of the memory locations referenced in the instruction.*

Arithmetic/Logic Unit (ALU)

- Performs the math
- Generally does the work during the Instruction Execute step of the Cycle
- A circuit in the ALU can add two number
- There are also circuits for multiplying, comparing, 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)
- When the ALU completes the operation, Return Result/Store step moves the answer from the ALU to the destination memory address specified in the instruction

Input Unit and Output Unit (I/O)

- The wires and circuits through which information moves into and out of a computer
- The *peripherals*: Connect to the computer input/output ports. They are not considered part of the computer, but specialized gadgets that encode or decode information between the computer and the physical world.

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

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.

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 part of the computer. It is called 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.

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.

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

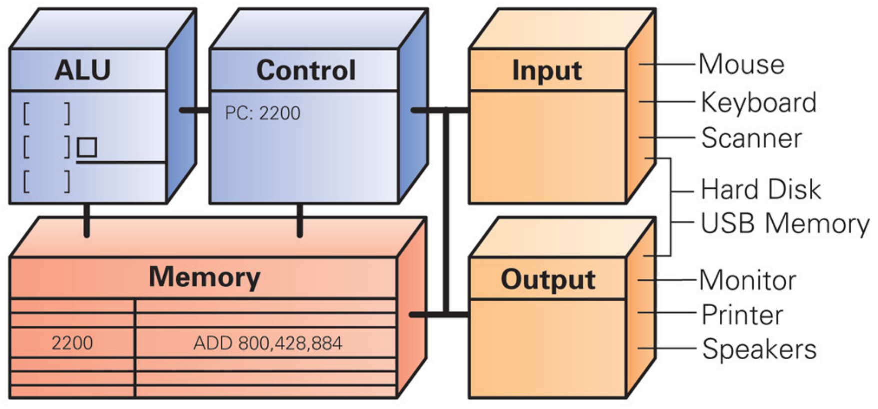


Figure 9.5. Computer before executing an ADD instruction.

Instruction Interpretation (cont'd)

- Execution begins by moving instruction at the address given by the PC from memory to control unit

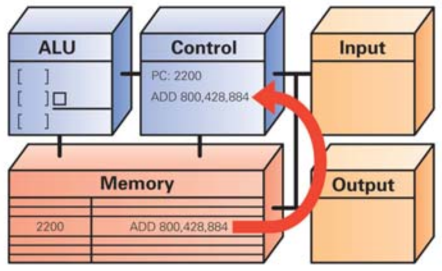


Figure 9.6. Instruction Fetch: Move instruction from memory to the control unit.

Instruction Interpretation (cont'd)

- Bits of instruction are placed into the decoder circuit of the CU
- Once instruction is fetched, the PC can be readied for fetching the next instruction

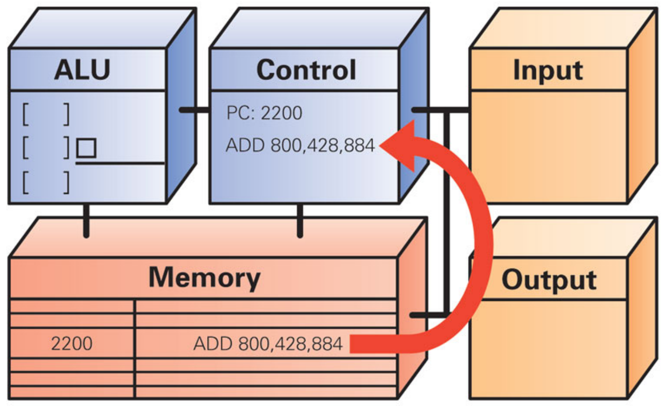


Figure 9.6. Instruction Fetch: Move instruction from memory to the control unit.

Instruction Interpretation (cont'd)

- In Instruction Decode step, ALU is set up for the operation
- Decoder will find the memory address of the instruction's data (source operands)
  - Most instructions operate on two data values stored in memory (like ADD), so most instructions have addresses for two source operands
  - These addresses are passed to the circuit that fetches them from memory during the next step, Data Fetch
- Decoder finds destination address for the Result Return step, and places it in RR circuit
- Decoder determines what operation the ALU will perform, and sets it up appropriately

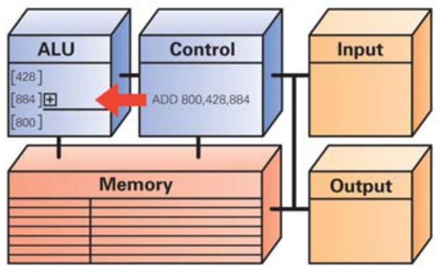


Figure 9.7. Instruction Decode: Pull apart the instruction, set up the operation in the ALU, and compute the source and destination operand addresses.

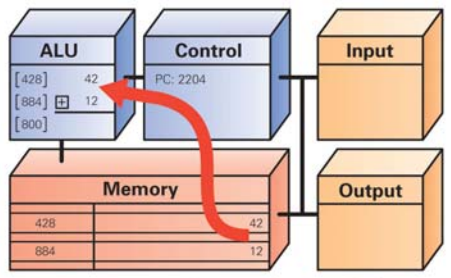


Figure 9.8. Data Fetch: Move the operands from memory to the ALU.

Instruction Interpretation (cont'd)

- Instruction Execution: The actual computation is performed. For ADD instruction, the addition circuit adds the two source operands together to produce their sum

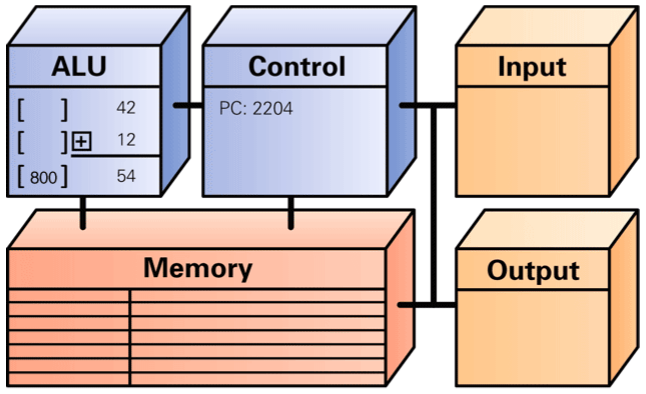


Figure 9.9 Instruction Execute: Compute the result of the operation in the ALU.

Instruction Interpretation (cont'd)

- Result Return: result of execution is returned to the memory location specified by the destination address.
- Once the result is returned, the cycle begins again.

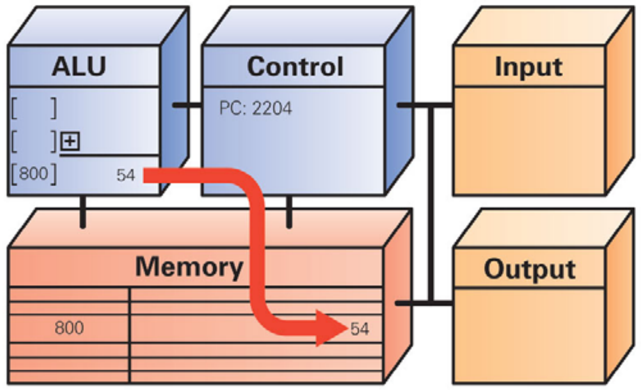


Figure 9.10. Result Return: Store the result from the ALU into the memory at the destination address.

Many, Many Simple Operations

- Computers can only perform about 100 different instructions
  - About 20 different kinds of operations (different instructions are needed for adding bytes, words, decimal numbers, etc.)
- Everything computers do must be reduced to some combination of these primitive, hardwired instructions

Examples of Other Instructions

- Besides ADD, MULT (multiply) and DIV (divide), other instructions include:
  - Shift the bits of a word to the left or right, filling the emptied places with zeros and throwing away bits that fall off the end
  - Compute logical AND (test if pairs of bits are both true), and logical OR (test if at least one of two bits is true)
  - Test if a bit is zero or non-zero, and jump to new set of instructions based on outcome
  - Move information around in memory
  - Sense signals from input/output devices

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

1000^1	kilo-	$1024^1 = 2^{10} = 1{,}024$	milli-	1000^{-1}
1000^2	mega-	$1024^2 = 2^{20} = 1{,}048{,}576$	micro-	1000^{-2}
1000^3	giga-	$1024^3 = 2^{30} = 1{,}073{,}741{,}824$	nano-	1000^{-3}
1000^4	tera-	$1024^4 = 2^{40} = 1{,}099{,}511{,}627{,}776$	pico-	1000^{-4}
1000^5	peta-	$1024^5 = 2^{50} = 1{,}125{,}899{,}906{,}842{,}624$	femto-	1000^{-5}
1000^6	exa-	$1024^6 = 2^{60} = 1{,}152{,}921{,}504{,}606{,}876{,}976$	atto-	1000^{-6}
1000^7	zetta-	$1024^7 = 2^{70} = 1{,}180{,}591{,}620{,}717{,}411{,}303{,}424$	zepto-	1000^{-7}
1000^8	yotta-	$1024^8 = 2^{80} = 1{,}208{,}925{,}819{,}614{,}629{,}174{,}706{,}176$	yocto-	1000^{-8}

Figure 9.11 Standard prefixes from the Système International (SI) convention on scientific measurements. Generally a prefix refers to a power of 1000, except when the quantity (for example, memory) is counted in binary; for binary quantities the prefix refers to a power of 1024, which is 2^{10}.

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

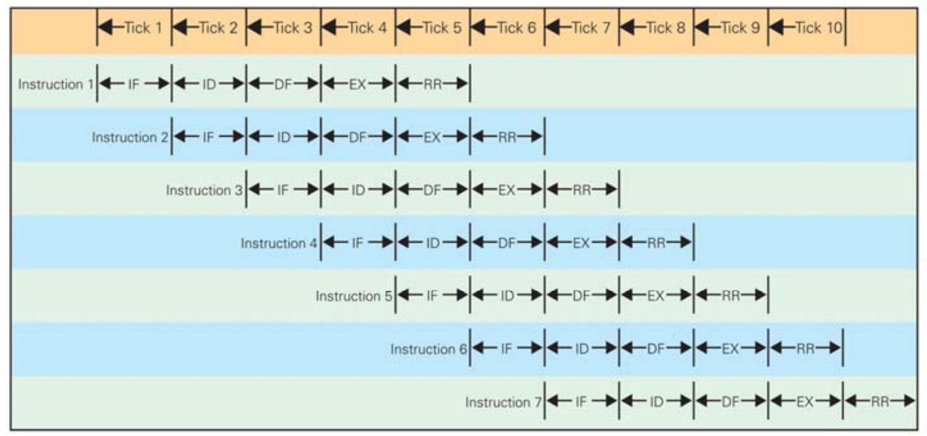


Figure 9.12. *Schematic diagram of a pipelined Fetch/Execute Cycle. On each tick, the IF circuit starts a new instruction, and then passes it along to the ID (Instruction Decode) unit; the ID unit works on the instruction it receives, and when it finishes, it passes it along to the DF (Data Fetch) circuit, and so on. When the pipeline is filled, five instructions are in progress at once, and one instruction is finished on each clock tick, making the computer appear to be running at one instruction per tick.*

Software

- A computer's view of software
  - Sees *binary object file*, a long sequence of 4-byte words (0's and 1's)
- *Assembly language*
  - Alternative form of machine language using letters and normal numbers so people can understand it
  - Computer scans assemble code, as it encounters words it looks them up in a table to convert to binary, converts numbers to binary, then assembles the binary pieces into an instruction

Software (cont'd)

- *High-level programming languages*
  - Most modern software is written in high-level notation, which is then *compiled* (translated) into assembly language, which is then assembled into binary
  - Have special statement forms to help programmers give complicated instructions
    - Example: Three-part if statement
      - Yes/no question to test
      - Instructions to operate if test is true
      - Instructions to operate if test is false

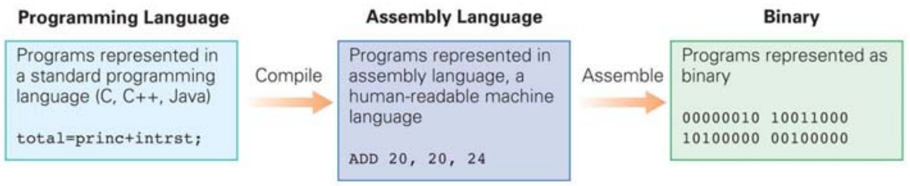
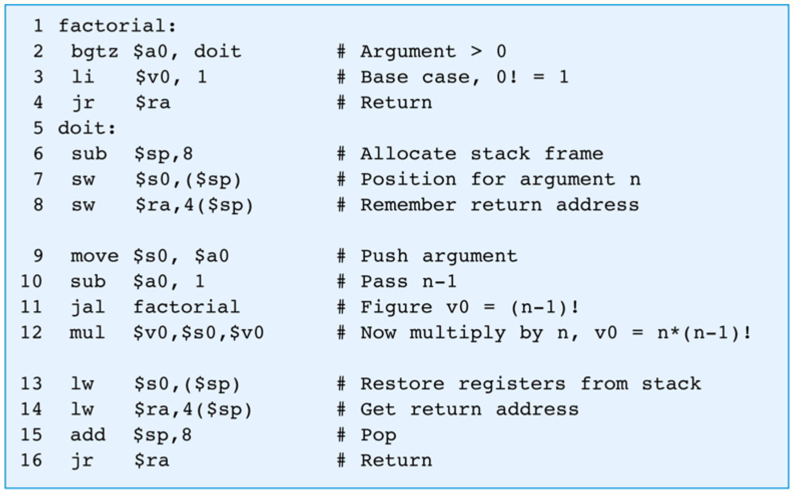

Programming Language
Programs represented in a standard programming language (C, C++, Java)
total=princ+intrst;
Compile
Assembly Language
Programs represented in assembly language, a human-readable machine language
ADD 20, 20, 24
Assemble
Binary
Programs represented as binary
00000010 10011000
10100000 00100000

Figure 9.13. The three primary forms of encoding software: programming language, assembly language, and binary machine language.

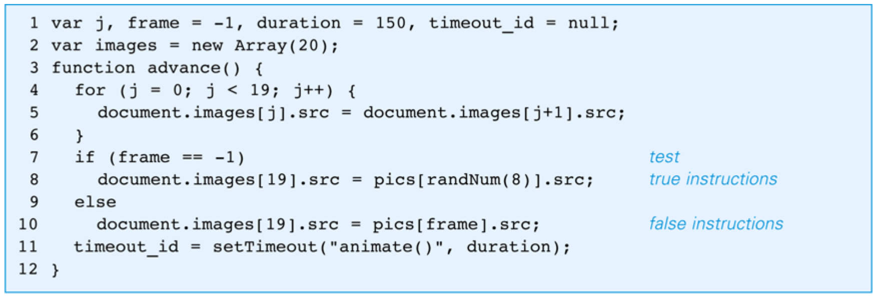


```
factorial:
 bgtz $a0, doit        # Argument > 0
 li   $v0, 1           # Base case, 0! = 1
 jr   $ra              # Return
doit:
 sub  $sp,8            # Allocate stack frame
 sw   $s0,($sp)        # Position for argument n
 sw   $ra,4($sp)       # Remember return address

 move $s0, $a0         # Push argument
 sub  $a0, 1           # Pass n-1
 jal  factorial        # Figure v0 = (n-1)!
 mul  $v0,$s0,$v0      # Now multiply by n, v0 = n*(n-1)!

 lw   $s0,($sp)        # Restore registers from stack
 lw   $ra,4($sp)       # Get return address
 add  $sp,8            # Pop
 jr   $ra              # Return
```

Figure 9.14 An assembly language program to compute *n*! for the MIPS computer.



```
var j, frame = -1, duration = 150, timeout_id = null;
var images = new Array(20);
function advance() {
  for (j = 0; j < 19; j++) {
    document.images[j].src = document.images[j+1].src;
  }
  if (frame == -1)                                          test
    document.images[19].src = pics[randNum(8)].src;         true instructions
  else
    document.images[19].src = pics[frame].src;              false instructions
  timeout_id = setTimeout("animate()", duration);
}
```

Figure 9.15 A fragment of program text written in a high-level language.

Operating Systems

- Basic operations that are necessary for the effective use of computer, but are not built into the hardware
- Three most widely used Operating Systems:
  - Microsoft Windows
  - Apple's Mac OS X
  - Unix / Linux
- OS performs booting, memory management, device management, Internet connection, file management

Programming

- Programmers build on previously developed software to make their jobs easier
- Example: GUI Software
  - Frame around window, slider bars, buttons, pointers, etc. are packaged for programmers and given with OS

Integrated Circuits

- Miniaturization:
  - Clock speeds are so high because processor chips are so tiny (electrical signals can travel about 1 foot in a nanosecond)

Integrated Circuits

- Photolithography
  - Printing process. Instead of hand-wiring circuits together, photograph what is wanted and etch away the spaces
  - Regardless of how complicated the wiring, cost and amount of work are the same

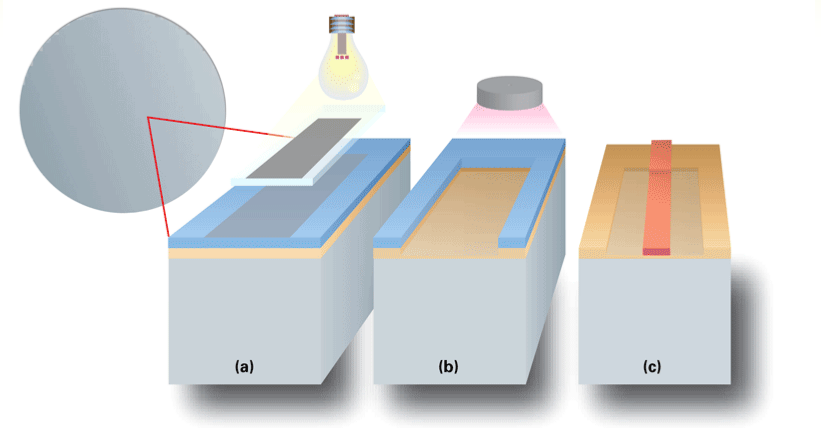


Figure 9.16 Early steps in the fabrication process. (a) A layer of photoresist (blue) is exposed to UV light through a pattern mask (light blue), hardening the exposed areas; (b) after washing away the unexposed photoresist, hot gases etch away (nearly all of) the exposed layer; (c) the remaining resist is washed away and other layers are created by repeating the patterning and etching processes. In later stages of the fabrication process, (d) "impurities" (green) such as

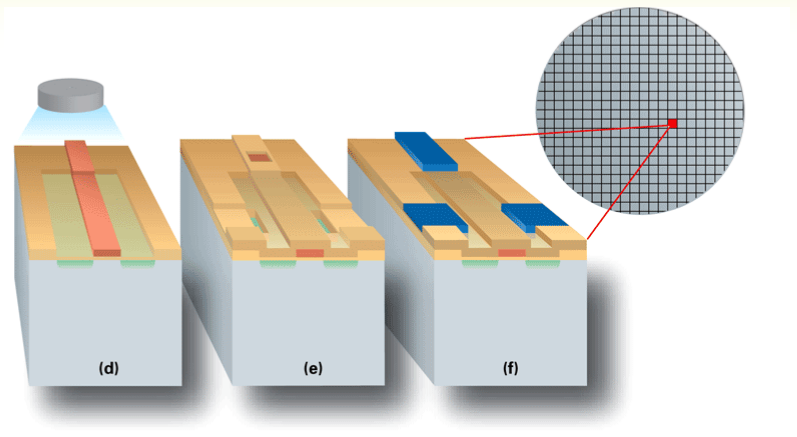


Figure 9.16 *Continued*
boron are diffused into the silicon surface in a process called doping, which improves the availability of electrons in this region of the silicon. (e) After additional layering, etching exposes contact points for metal wires, and (f) a metal (dark blue) such as aluminum is deposited creating "wires" to connect to other transistors. Millions of such transistors form a computer chip occupying a small square on the final fabricated wafer.

How Semi-conductor Technology Works

- Integration:
  - Active components and the wires that connect them are all made together of similar materials in a single process
  - Saves space and produces monolithic part for the whole system, which is more reliable
- Silicon is a semi-conductor—sometimes it conducts electricity, sometimes not
  - Ability to control when semi-conductor conducts is the main tool in computer construction

The On-Again, Off-Again Behavior of Silicon

- A circuit is set to compute x and y for any logical values x and y
- If x is true, the x circuit conducts electricity and a signal passes to the other end of the wire; if x is false, no signal passes
- Same process for y
- If both circuits conduct, x and y are true—logical AND has been computed

The Field Effect

- Controls the conductivity of the semiconductor
- Objects can become charged positively or negatively
  - Like charges repel each other, but opposites attract. This effect is called the *field effect*.

The Field Effect (cont'd)

- The gap between two wires is treated to improve its conducting and non-conducting properties
- This is called a *channel* (path for electricity to travel between the two wires)
- An insulator covers the channel
- A wire called the *gate* passes over the insulator
- The gate is separated from the channel by the insulator—does not make contact with the wires or the channel
- Electricity is not conducted between the two wires unless the channel is conducting

How Does the Channel Conduct?

- The silicon in the channel conducts electricity when it is in a charged field
  – Electrons are attracted or repelled in the silicon material
  – Charging the gate positively creates a field over the channel that conducts electricity between the two wires

Transistors

- A connector between two wires that can be controlled to allow charge to flow between the two wires, or not
- We have described a MOS transistor (Metal Oxide Semiconductor)

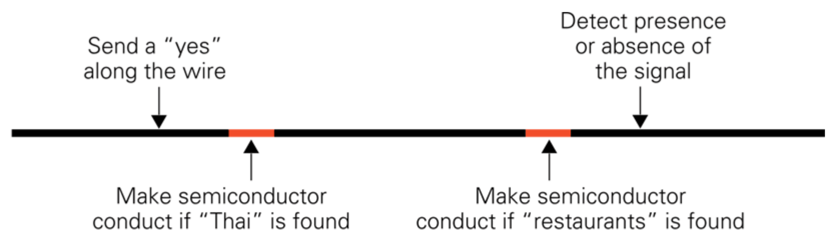

Send a "yes" along the wire
Detect presence or absence of the signal
Make semiconductor conduct if "Thai" is found
Make semiconductor conduct if "restaurants" is found

Figure 9.17 Computing `Thai AND restaurants` using a semiconducting material.

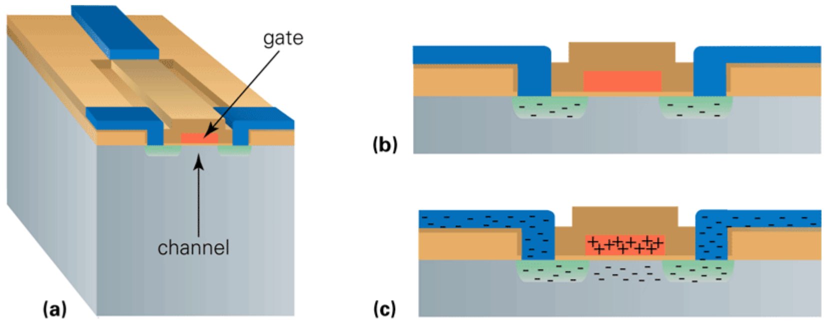


Figure 9.18 Operation of a field effect transistor. (a) Cross-section of the transistor of Figure 9.16(f). (b) The gate (red) is neutral and the channel, the region in the silicon below the gate, does not conduct, isolating the wires (blue); (c) charging the gate causes the channel to conduct, connecting the wires.

Combining the Ideas

- Put all above ideas together:
  - Start with information processing task
  - Task is performed by application, implemented as part of a large program in a high-level language like C or Java
  - Program performs specific operations; standard operations like print or save are done by OS
  - Program's commands are compiled into assembly language instructions
  - Assembly instructions are translated into binary code
  - Binary instructions are stored on hard disk (secondary memory)
  - Application instructions move into RAM (primary memory)

Combining the Ideas (cont'd)

- Fetch/Execute Cycle executes the instructions
- All the computer's instructions are performed by the ALU circuits, using the transistor model previously described, under the control of the Control Unit

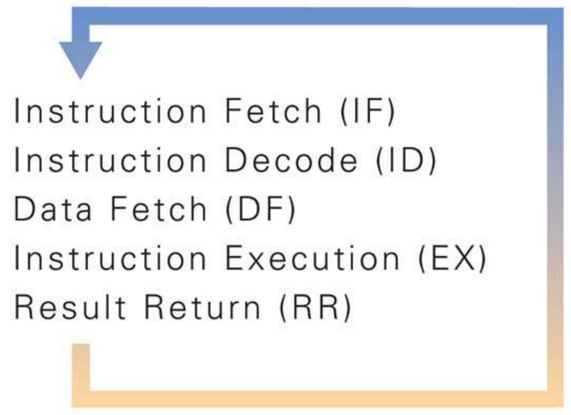


Figure 9.1. The Fetch/Execute Cycle.