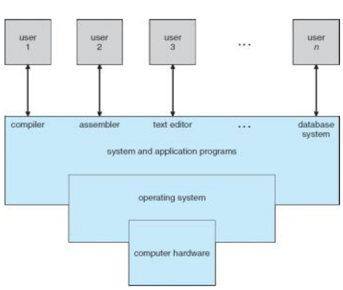
Introduction, Data Representation I

CMSC 313
Sections 01, 02

[Review of Syllabus, Web pages]

2

Four Components of a Computer System



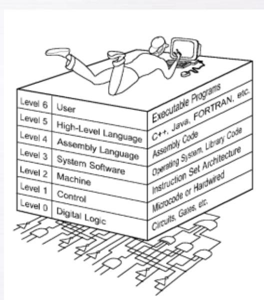
Adapted fm Silberschatz, Galvin & Gagne, 2013

1.6 The Computer Level Hierarchy

- Computers consist of many things besides chips.
- Before a computer can do anything worthwhile, it must also use software.
- Writing complex programs requires a “divide and conquer” approach, where each program module solves a smaller problem.
- Complex computer systems employ a similar technique through a series of virtual machine layers.

1.6 The Computer Level Hierarchy

- Each virtual machine layer is an abstraction of the level below it.
- The machines at each level execute their own particular instructions, calling upon machines at lower levels to perform tasks as required.
- Computer circuits ultimately carry out the work.



1.6 The Computer Level Hierarchy

- Level 6: The User Level
  - Program execution and user interface level.
  - The level with which we are most familiar.
- Level 5: High-Level Language Level
  - The level with which we interact when we write programs in languages such as C, Pascal, Lisp, and Java.

1.6 The Computer Level Hierarchy

- Level 4: Assembly Language Level
  - Acts upon assembly language produced from Level 5, as well as instructions programmed directly at this level.
- Level 3: System Software Level
  - Controls executing processes on the system.
  - Protects system resources.
  - Assembly language instructions often pass through Level 3 without modification.

7

1.6 The Computer Level Hierarchy

- Level 2: Machine Level
  - Also known as the Instruction Set Architecture (ISA) Level.
  - Consists of instructions that are particular to the architecture of the machine.
  - Programs written in machine language need no compilers, interpreters, or assemblers.

8

1.6 The Computer Level Hierarchy

- Level 1: Control Level
  - A *control unit* decodes and executes instructions and moves data through the system.
  - Control units can be *microprogrammed* or *hardwired*.
  - A microprogram is a program written in a low-level language that is implemented by the hardware.
  - Hardwired control units consist of hardware that directly executes machine instructions.

9

1.6 The Computer Level Hierarchy

- Level 0: Digital Logic Level
  - This level is where we find digital circuits (the chips).
  - Digital circuits consist of gates and wires.
  - These components implement the mathematical logic of all other levels.

10

1.8 The von Neumann Model

- On the ENIAC, all programming was done at the digital logic level.
- Programming the computer involved moving plugs and wires.
- A different hardware configuration was needed to solve every unique problem type.

Configuring the ENIAC to solve a "simple" problem required many days labor by skilled technicians.

11

1.8 The von Neumann Model

- Inventors of the ENIAC, John Mauchley and J. Presper Eckert, conceived of a computer that could store instructions in memory.
- The invention of this idea has since been ascribed to a mathematician, John von Neumann, who was a contemporary of Mauchley and Eckert.
- Stored-program computers have become known as von Neumann Architecture systems.

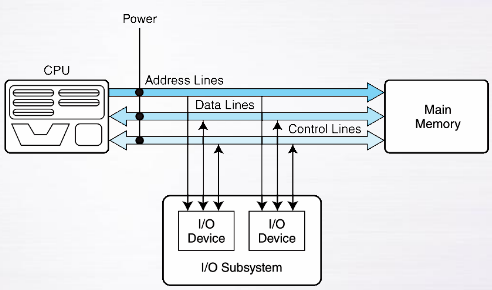
12

1.8 The von Neumann Model

- Today's stored-program computers have the following characteristics:
  - Three hardware systems:
    - A central processing unit (CPU)
    - A main memory system
    - An I/O system
  - The capacity to carry out sequential instruction processing.
  - A single data path between the CPU and main memory.
    - This single path is known as the *von Neumann bottleneck*.

13

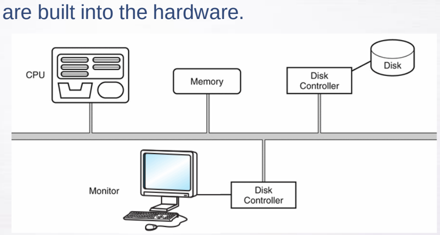
4.3 The Bus



14

4.3 The Bus

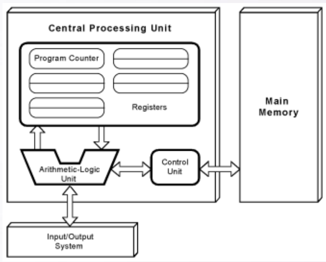
- A multipoint bus is shown below.
- Because a multipoint bus is a shared resource, access to it is controlled through protocols, which are built into the hardware.



15

1.8 The von Neumann Model

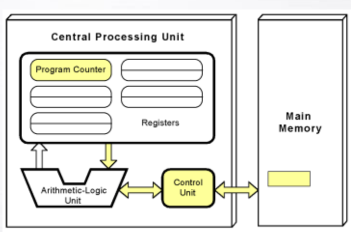
- This is a general depiction of a von Neumann system:
- These computers employ a fetch-decode-execute cycle to run programs as follows . . .



16

1.8 The von Neumann Model

- **The control unit fetches the next instruction from memory using the program counter to determine where the instruction is located.**

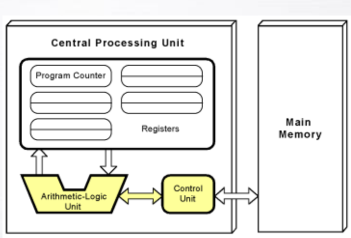


17

© Odua Images/ShutterStock, Inc. Copyright © 2014 by Jones & Bartlett Learning, LLC an Ascend Learning Company www.jblearning.com

1.8 The von Neumann Model

- **The instruction is decoded into a language that the ALU can understand.**

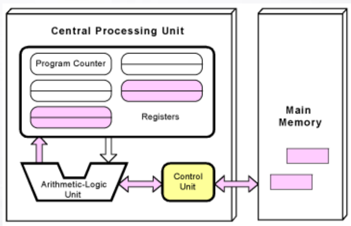


18

© Odua Images/ShutterStock, Inc. Copyright © 2014 by Jones & Bartlett Learning, LLC an Ascend Learning Company www.jblearning.com

1.8 The von Neumann Model

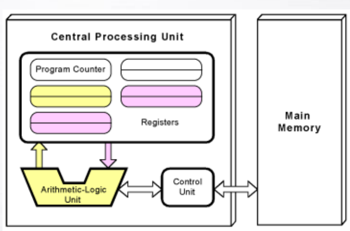
- Any data operands required to execute the instruction are fetched from memory and placed into registers within the CPU.



19

1.8 The von Neumann Model

- The ALU executes the instruction and places results in registers or memory.



20

BASE CONVERSION

21

2.1 Introduction

- A *bit* is the most basic unit of information in a computer.
  - It is a state of "on" or "off" in a digital circuit.
  - Sometimes these states are "high" or "low" voltage instead of "on" or "off.."
- A *byte* is a group of eight bits.
  - A byte is the smallest possible *addressable* unit of computer storage.
  - The term, "addressable," means that a particular byte can be retrieved according to its location in memory.

22

2.1 Introduction

- A *word* is a contiguous group of bytes.
  - Words can be any number of bits or bytes.
  - Word sizes of 16, 32, or 64 bits are most common.
  - In a word-addressable system, a word is the smallest addressable unit of storage.
- A group of four bits is called a *nibble*.
  - Bytes, therefore, consist of two nibbles: a "high-order nibble," and a "low-order" nibble.

23

2.2 Positional Numbering Systems

- Bytes store numbers using the position of each bit to represent a power of 2.
  - The binary system is also called the base-2 system.
  - Our decimal system is the base-10 system. It uses powers of 10 for each position in a number.
  - Any integer quantity can be represented exactly using any base (or *radix*).

24

2.2 Positional Numbering Systems

- The decimal number 947 in powers of 10 is:

$$9 \times 10^2 + 4 \times 10^1 + 7 \times 10^0$$

- The decimal number 5836.47 in powers of 10 is:

$$5 \times 10^3 + 8 \times 10^2 + 3 \times 10^1 + 6 \times 10^0 + 4 \times 10^{-1} + 7 \times 10^{-2}$$

25

2.2 Positional Numbering Systems

- The binary number 11001 in powers of 2 is:

$$1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$
$$= 16 + 8 + 0 + 0 + 1 = 25$$

- When the radix of a number is something other than 10, the base is denoted by a subscript.
  - Sometimes, the subscript 10 is added for emphasis:

$$11001_2 = 25_{10}$$

26

2.3 Converting Between Bases

- Because binary numbers are the basis for all data representation in digital computer systems, it is important that you become proficient with this radix system.
- Your knowledge of the binary numbering system will enable you to understand the operation of all computer components as well as the design of instruction set architectures.

27

2.3 Converting Between Bases

- In an earlier slide, we said that every integer value can be represented exactly using any radix system.
- There are two methods for radix conversion: the subtraction method and the division remainder method.
- The subtraction method is more intuitive, but cumbersome. It does, however reinforce the ideas behind radix mathematics.

28

2.3 Converting Between Bases

- **Suppose we want to convert the decimal number 190 to base 3.**
  - We know that $3^5 = 243$ so our result will be less than six digits wide. The largest power of 3 that we need is therefore $3^4 = 81$, and $81 \times 2 = 162$.
  - Write down the 2 and subtract 162 from 190, giving 28.

$$\begin{array}{r l} 190 & \\ -\ 162 & = 3^4 \times 2 \\ \hline 28 & \end{array}$$

29

2.3 Converting Between Bases

- **Converting 190 to base 3...**
  - The next power of 3 is $3^3 = 27$. We'll need one of these, so we subtract 27 and write down the numeral 1 in our result.
  - The next power of 3, $3^2 = 9$, is too large, but we have to assign a placeholder of zero and carry down the 1.

$$\begin{array}{r l} 190 & \\ -\ 162 & = 3^4 \times 2 \\ \hline 28 & \\ -\ 27 & = 3^3 \times 1 \\ \hline 1 & \\ -\ 0 & = 3^2 \times 0 \\ \hline 1 & \end{array}$$

30

2.3 Converting Between Bases

- **Converting 190 to base 3...**
  - $3^1 = 3$ is again too large, so we assign a zero placeholder.
  - The last power of 3, $3^0 = 1$, is our last choice, and it gives us a difference of zero.
  - Our result, reading from top to bottom is:

    $190_{10} = 21001_3$

$$
\begin{array}{rl}
 & 190 \\
- & 162 = 3^4 \times 2 \\
 & 28 \\
- & 27 = 3^3 \times 1 \\
 & 1 \\
- & 0 = 3^2 \times 0 \\
 & 1 \\
- & 0 = 3^1 \times 0 \\
 & 1 \\
- & 1 = 3^0 \times 1 \\
 & 0
\end{array}
$$

31

2.3 Converting Between Bases

- Another method of converting integers from decimal to some other radix uses division.
- This method is mechanical and easy.
- It employs the idea that successive division by a base is equivalent to successive subtraction by powers of the base.
- Let's use the division remainder method to again convert 190 in decimal to base 3.

32

2.3 Converting Between Bases

- **Converting 190 to base 3...**
  - First we take the number that we wish to convert and divide it by the radix in which we want to express our result.
  - In this case, 3 divides 190 63 times, with a remainder of 1.
  - Record the quotient and the remainder.

$$
\begin{array}{r|rl}
3 & 190 & 1 \\
\hline
 & 63 &
\end{array}
$$

33

2.3 Converting Between Bases

- **Converting 190 to base 3...**
  - 63 is evenly divisible by 3.
  - Our remainder is zero, and the quotient is 21.

3	190	1
3	63	0
	21	

34

2.3 Converting Between Bases

- **Converting 190 to base 3...**
  - Continue in this way until the quotient is zero.
  - In the final calculation, we note that 3 divides 2 zero times with a remainder of 2.
  - Our result, reading from bottom to top is:

    $190_{10} = 21001_3$

3	190	1
3	63	0
3	21	0
3	7	1
3	2	2
	0	

35

2.3 Converting Between Bases

- The binary numbering system is the most important radix system for digital computers.
- However, it is difficult to read long strings of binary numbers -- and even a modestly-sized decimal number becomes a very long binary number.
  - For example: $11010100011011_2 = 13595_{10}$
- For compactness and ease of reading, binary values are usually expressed using the hexadecimal, or base-16, numbering system.

36

2.3 Converting Between Bases

- The hexadecimal numbering system uses the numerals 0 through 9 and the letters A through F.
  - The decimal number 12 is C_{16}.
  - The decimal number 26 is $1A_{16}$.
- It is easy to convert between base 16 and base 2, because $16 = 2^4$.
- Thus, to convert from binary to hexadecimal, all we need to do is group the binary digits into groups of four.

A group of four binary digits is called a hextet

37

2.3 Converting Between Bases

- Using groups of hextets, the binary number 11010100011011_2 (= 13595_{10}) in hexadecimal is:

0011	0101	0001	1011
3	5	1	B

If the number of bits is not a multiple of 4, pad on the left with zeros.

- Octal (base 8) values are derived from binary by using groups of three bits ($8 = 2^3$):

011	010	100	011	011
3	2	4	3	3

Octal was very useful when computers used six-bit words.

38

2.3 Converting Between Bases

- Fractional values can be approximated in all base systems.
- Unlike integer values, fractions do not necessarily have exact representations under all radices.
- The quantity ½ is exactly representable in the binary and decimal systems, but is not in the ternary (base 3) numbering system.

39

2.3 Converting Between Bases

- Fractional decimal values have nonzero digits to the right of the decimal point.
- Fractional values of other radix systems have nonzero digits to the right of the *radix point*.
- Numerals to the right of a radix point represent negative powers of the radix:

$$0.47_{10} = 4 \times 10^{-1} + 7 \times 10^{-2}$$
$$0.11_2 = 1 \times 2^{-1} + 1 \times 2^{-2}$$
$$= \tfrac{1}{2} + \tfrac{1}{4}$$
$$= 0.5 + 0.25 = 0.75$$

40

2.3 Converting Between Bases

- As with whole-number conversions, you can use either of two methods: a subtraction method or an easy multiplication method.
- The subtraction method for fractions is identical to the subtraction method for whole numbers. Instead of subtracting positive powers of the target radix, we subtract negative powers of the radix.
- We always start with the largest value first, n^{-1}, where n is our radix, and work our way along using larger negative exponents.

41

2.3 Converting Between Bases

- **The calculation to the right is an example of using the subtraction method to convert the decimal 0.8125 to binary.**
  - Our result, reading from top to bottom is:

    $0.8125_{10} = 0.1101_2$
  - Of course, this method works with any base, not just binary.

```
  0.8125
- 0.5000  = 2^-1 × 1
  0.3125
- 0.2500  = 2^-2 × 1
  0.0625
-      0  = 2^-3 × 0
  0.0625
- 0.0625  = 2^-4 × 1
       0
```

42

2.3 Converting Between Bases

- **Using the multiplication method to convert the decimal 0.8125 to binary, we multiply by the radix 2.**
  - The first product carries into the units place.

```
  .8125
  ×   2
 1.6250
```


2.3 Converting Between Bases

- **Converting 0.8125 to binary . . .**
  - Ignoring the value in the units place at each step, continue multiplying each fractional part by the radix.

```
  .8125
  ×   2
 1.6250

  .6250
  ×   2
 1.2500

  .2500
  ×   2
 0.5000
```


2.3 Converting Between Bases

- **Converting 0.8125 to binary . . .**
  - You are finished when the product is zero, or until you have reached the desired number of binary places.
  - Our result, reading from top to bottom is:

    $0.8125_{10} = 0.1101_2$
  - This method also works with any base. Just use the target radix as the multiplier.

```
  .8125
  ×   2
 1.6250

  .6250
  ×   2
 1.2500

  .2500
  ×   2
 0.5000

  .5000
  ×   2
 1.0000
```

Convert Base 6 to Base 10

123.45_6 = ???.?? $_{10}$
$123_6 = 1*6^2{}_{10}$ [$1*36_{10}$] +
$2*6^1{}_{10}$ [$2*6_{10}$] +
$3*6^0{}_{10}$ [$3*1_{10}$] =
51_{10}
$0.45_6 = 4*6^{-1}{}_{10}$ [$4*1/6_{10}$] +
$5*6^{-2}{}_{10}$ [$5*1/36_{10}$] =
$.80555\ldots_{10}$
$123.45_6 = 51.80555\ldots_{10}$

46

Convert Base 10 to Base 6

754.94_{10} = 3254.5 35012 35012 35012...$_6$
754 / 6 = 125 remainder 4
125 / 6 = 20 remainder 5
20 / 6 = 3 remainder 2
3 / 6 = 0 remainder 3

$3254_6 = 3 \times 216_{10} + 2 \times 36_{10} + 5 \times 6_{10} + 4 \times 1_{10}$
$= 754_{10}$

47

Convert Base 10 to Base 6

$.94_{10}$ = ???.???$_6$
0.94 x 6 = 5.64 --> 5
0.64 x 6 = 3.84 --> 3
0.84 x 6 = 5.04 --> 5
0.04 x 6 = 0.24 --> 0
0.24 x 6 = 1.44 --> 1
0.44 x 6 = 2.64 --> 2
0.64 x 6 = 3.84 --> 3

0.94_{10} = 0.5 35012 35012 35012...$_6$
$5/6 + 3/36 + 5/216 + 0 + 1/6^5 + 2/6^6 = 0.939986282\ldots_{10}$

48