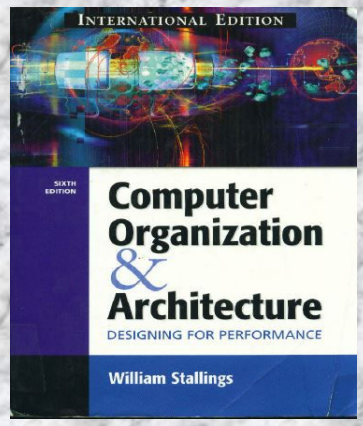
Lab1 Introduction

Computer Organization & Architecture

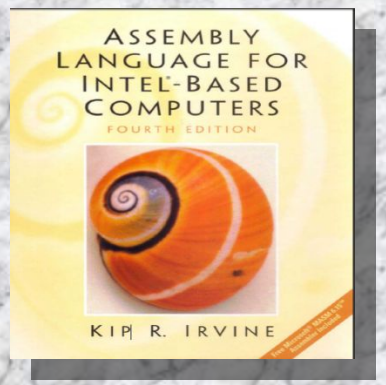
Nedaa Hussein Ahmed
nh1179@fayoum.edu.eg

Course information

"Computer Organization And Architecture", 6th edition, by William Stallings.



" Assembly language for INTEL-based computers", 4th edition , by KIP R. I RV I N E

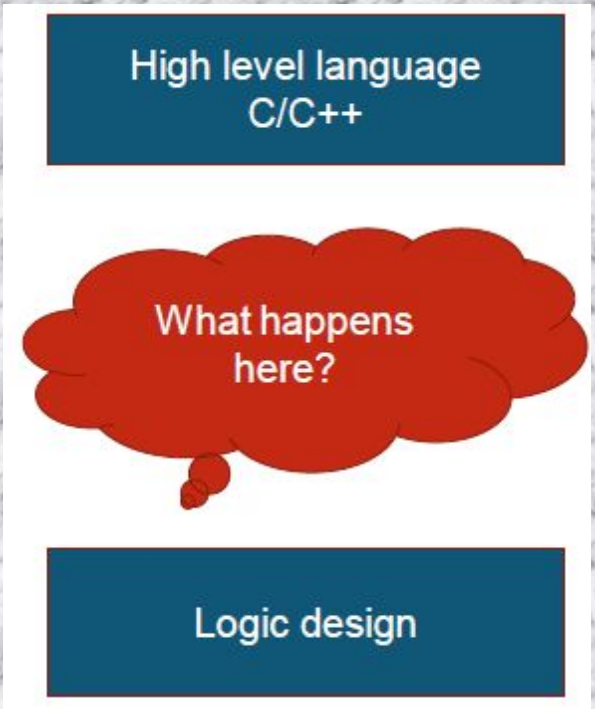


Emu8086 → 8086 microprocessor emulator version 4.08.

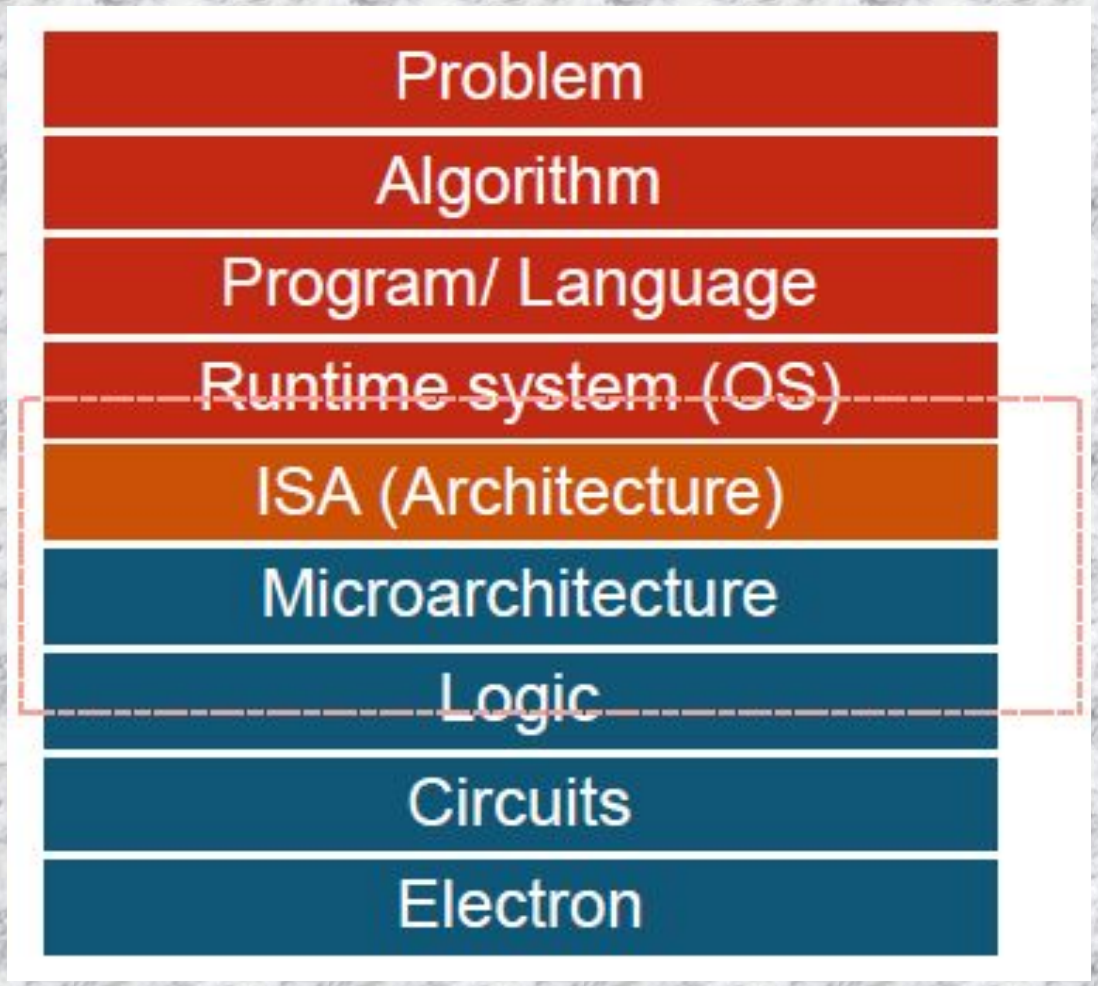
MDA 8086 trainer kit.

Introduction

A program written by a high level language is translated into an assembly language before it can be executed by the computer H/W.



Introduction cont...



Machine Languages

- In the earliest days of computers, the only programming language s available were **machine languages.**
- It was made of streams of 0s and 1s.
- The only language understood by a computer is machine language.

Example:

Let us see a machine language program that do the following steps:

- Read two integers,
- Add them and,
- Print the result.

Machines Language cont...

Use eleven lines of code to do the required program, each line is 16 bits.

Code in machine language to add two integers

Hexadecimal	*Code in machine language*			
$(1FEF)_{16}$	0001	1111	1110	1111
$(240F)_{16}$	0010	0100	0000	1111
$(1FEF)_{16}$	0001	1111	1110	1111
$(241F)_{16}$	0010	0100	0001	1111
$(1040)_{16}$	0001	0000	0100	0000
$(1141)_{16}$	0001	0001	0100	0001
$(3201)_{16}$	0011	0010	0000	0001
$(2422)_{16}$	0010	0100	0010	0010
$(1F42)_{16}$	0001	1111	0100	0010
$(2FFF)_{16}$	0010	1111	1111	1111
$(0000)_{16}$	0000	0000	0000	0000

Assembly Languages

- The next evolution in programming came with the idea of replacing **binary code** for **instruction** and **addresses** with **symbols**.
- Because they used symbols, these languages were first known as **symbolic languages**.
- The set of these symbolic languages were later referred to as **assembly languages**.
- It provides direct access to a computer's hardware, making it necessary for you to understand a great deal about your comput er's architecture and operating system.

This course focuses on **programming Intel microprocessors** by using **Assembly language.**

Let's do the same example….

Assembly Languages cont...

Code in assembly language to add two integers

Code in assembly language				*Description*
LOAD	**RF**	**Keyboard**		Load from keyboard controller to register F
STORE	**Number1**	**RF**		Store register F into Number1
LOAD	**RF**	**Keyboard**		Load from keyboard controller to register F
STORE	**Number2**	**RF**		Store register F into Number2
LOAD	**R0**	**Number1**		Load Number1 into register 0
LOAD	**R1**	**Number2**		Load Number2 into register 1
ADDI	**R2**	**R0**	**R1**	Add registers 0 and 1 with result in register 2
STORE	**Result**	**R2**		Store register 2 into Result
LOAD	**RF**	**Result**		Load Result into register F
STORE	**Monitor**	**RF**		Store register F into monitor controller
HALT				Stop

High-Level Languages

- Working with symbolic languages was also very tedious, because each machine instruction had to be individually coded.
- The desire to improve programmer efficiency led to the development of high-level languages.
- Over the years, various languages, most notably BASIC, Pascal, C, C++ and Java, were developed.

Let's show the code for adding two integers as it would appear in the C++ language.

High-Level Languages cont...

Addition program in C++

```
/*      This program reads two integers from keyboard and prints their sum.
        Written by:
        Date:
*/
#include <iostream.h>
using namespace std;
int main (void)
{
        // Local Declarations
        int number1;
        int number2;
        int result;
        // Statements
        cin >> number1;
        cin >> number2;
        result = number1 + number2;
        cout << result;
        return 0;
} // main
```

Important Questions

➢What background should I have?

✓You should have completed a single college course or its equivalent in computer programming.

✓ Like c/c++, C#, Java..

➢Why learn computer architecture & assembly language?

✓Creating embedded system programs.

✓For games programmers who wants to takes full advantage of specific hardware features in a target system.

✓Gain an over all understanding of the interaction between the computer hardware, operating system, and application programs.

✓Using fast libraries (DLL) made in Assembly.

Assembler & Linker

- An assembler is a program that converts **source-code** programs from **assembly language** into **machine language**.
- Two of the most popular assemblers for the Intel family are **MASM** (Microsoft Assembler) and **TASM** (Borland Turbo Assembler).
- A linker combines **individual files** created by an assembler into a **single executable program**.
- A third program, called **a debugger** provides a way for a programmer to **trace** the execution of a program and examine the contents of memory.

Needed H/W & S/W

- **Hardware:**
  - ✓You need a computer with an Inte1386, Inte1486, or One of the Pentium processors.
- **Software:**
  - ✓OS: Windows, MS-DOS, or even Linux with DOS emulator.
  - ✓Editor: To write assembly code.
  - ✓Assembler: Like MASM
  - ✓Linker.
  - ✓Debugger: MASM supplies a good 16-bit debugger named CodeView.
- **How do C++ and Java relate to assembly language?**
  - ✓A single statement in C++ (Or Java) expands into multiple assembly language or machine instructions.

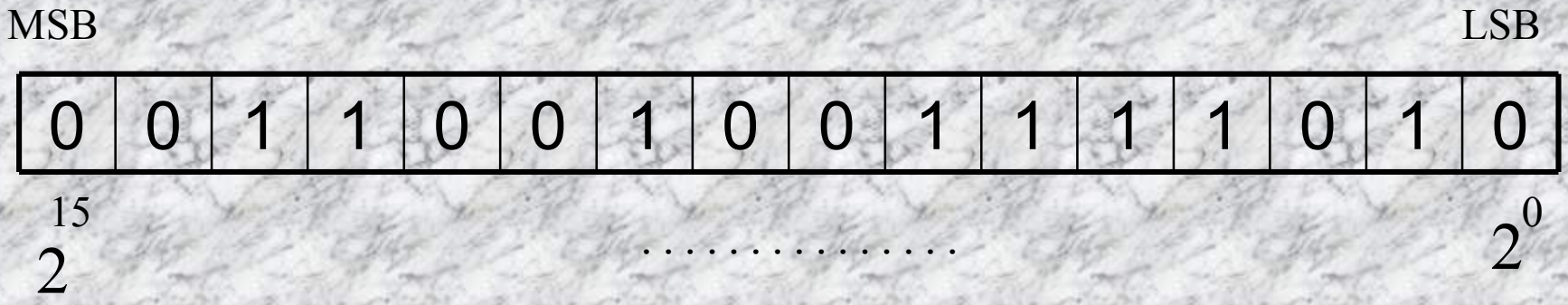
Numbering Systems

- To understand machine language and then assembly language we must first have a quick review to the previous knowledge of Number Systems.
- Number systems that we will work with are:
  - Binary → Base 2
  - Octal → Base 8
  - Decimal → Base 10
  - Hexadecimal → Base 16

Numbering Systems cont...

System	Base	Possible Digits
Binary	2	01
Octal	8	01234567
Decimal	10	0123456789
Hexadecimal	16	0 1 2 3 4 5 6 7 8 9 A BCDEF

1- Binary Numbers



- Base 2 system.
- Uses digits 0 &1.

2^n	Decimal Value
2^0	1
2^1	2
2^2	4
:	:
2^8	256
:	:
2^{15}	32768

1- Binary Numbers cont...

Translating unsigned binary integers to decimal:

- 8- bit Binary 00001001
- Decimal equivalence = $1*2^0+1*2^3 = 9$

Translating unsigned decimal integers to binary:

- Divide the decimal value by 2, saving each remainder as a binary digit. Binary of 37d = 100101b

Division	Quotient	Remainder
37/2	18	1 LSB
18/2	9	0
9/2	4	1
4/2	2	0
2/2	1	0
1/2	0	1 MSB

100101

1- Binary Numbers cont...

Convert 67 to its binary equivalent:

$67_{10} = x_2$

Step 1: 67 / 2 = 33 R 1 — *Divide 67 by 2. Record quotient in next row*

Step 2: 33 / 2 = 16 R 1 — *Again divide by 2; record quotient in next row*

Step 3: 16 / 2 = 8 R 0 — *Repeat again*

Step 4: 8 / 2 = 4 R 0 — *Repeat again*

Step 5: 4 / 2 = 2 R 0 — *Repeat again*

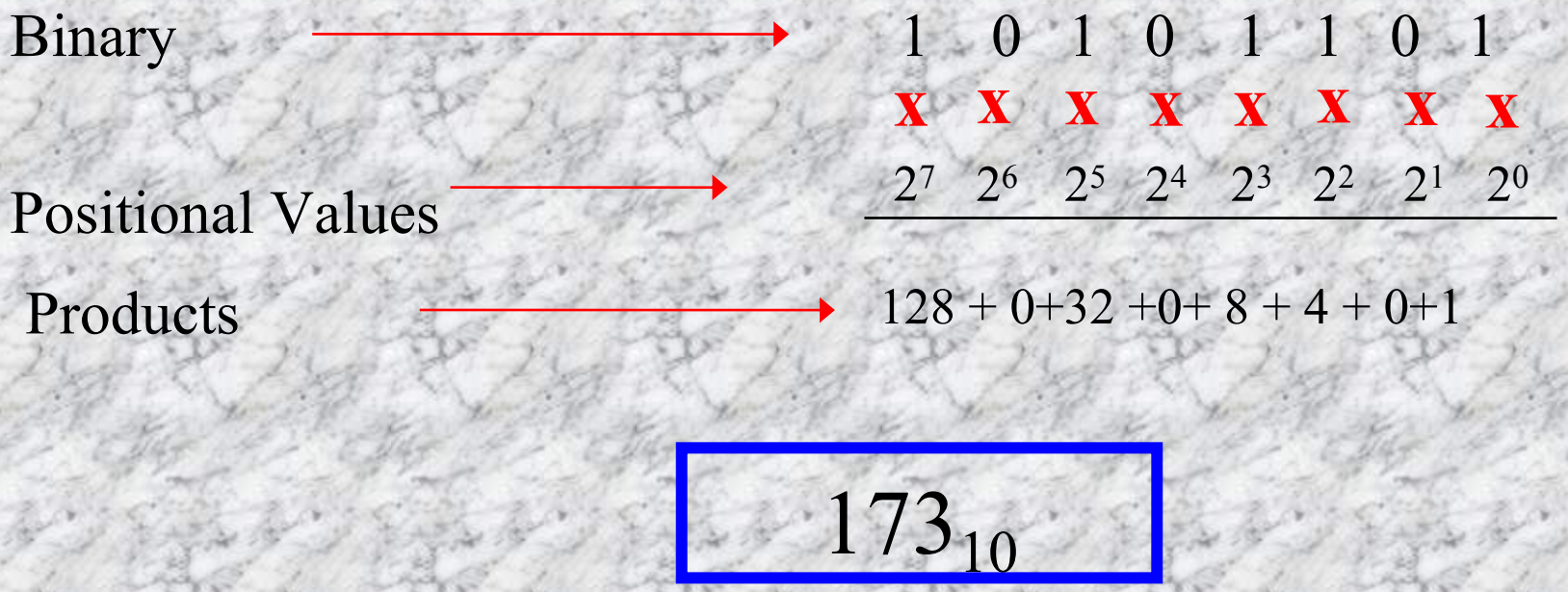
Step 6: 2 / 2 = 1 R 0 — *Repeat again*

Step 7: 1 / 2 = 0 R 1 — **STOP** *when quotient equals 0*

$1\,0\,0\,0\,0\,1\,1_2$

1- Binary Numbers cont...

Convert $(10101101)_2$ to its decimal equivalent:



173_{10}

Binary Addition

Rules:

- 0 + 0 = 0
- 0 + 1 = 1
- 1 + 0 = 1
- 1 + 1 = 10 (0 and Carry 1)

Example: add 12+7 convert to binary and then add

```
  1100
+ 0111
------
 10011
```

Complementary Arithmetic

- How do you represent a minus sign electronically in a computer?
- How can you represent it such that arithmetic operations are manageable?
- There are two types of compliments for each number base system.
  - ➢Have the r's complement
  - ➢Have the (r-1)'s complement
- For base 2 have 2's complement and 1's complement

1's complement:

➢Switch all 0's to 1's and 1's to 0's

Binary #	→	10110011
1's complement	→	01001100

Complementary Arithmetic

- **2's complement:**
  - **Step 1:** Find 1's complement of the number

Binary # → 11000110

1's complement → 00111001

  - **Step 2:** Add 1 to the 1's complement

$$\begin{array}{r} 00111001 \\ +\ 00000001 \\ \hline 00111010 \end{array}$$

Binary Subtraction

- We can deal with the second number as it is negative so we must find the 2's complement of it and convert the subtraction operation to summation.

- Now do the operation 4 – 6

4 → 0100 → 0100 → 0100

\- 6 → 0110 → 1001+1=1010 → +1010

1110 → 0001+1=- 0010

=-2

2- Octal Number System

- Also known as the **Base 8 System**
- Uses digits **0 - 7**
- Groups of three (binary) digits can be used to represent each octal digit
- Also uses multiplication and division algorithms for conversion to and from base 10

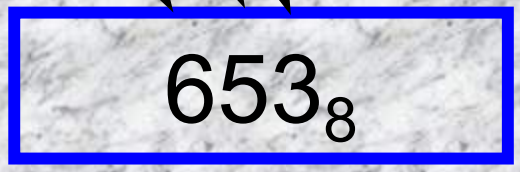
Decimal to Octal Conversion

Convert 427_{10} to its octal equivalent:

427 / 8 = 53 R= 3 Divide by 8; R is Reminder

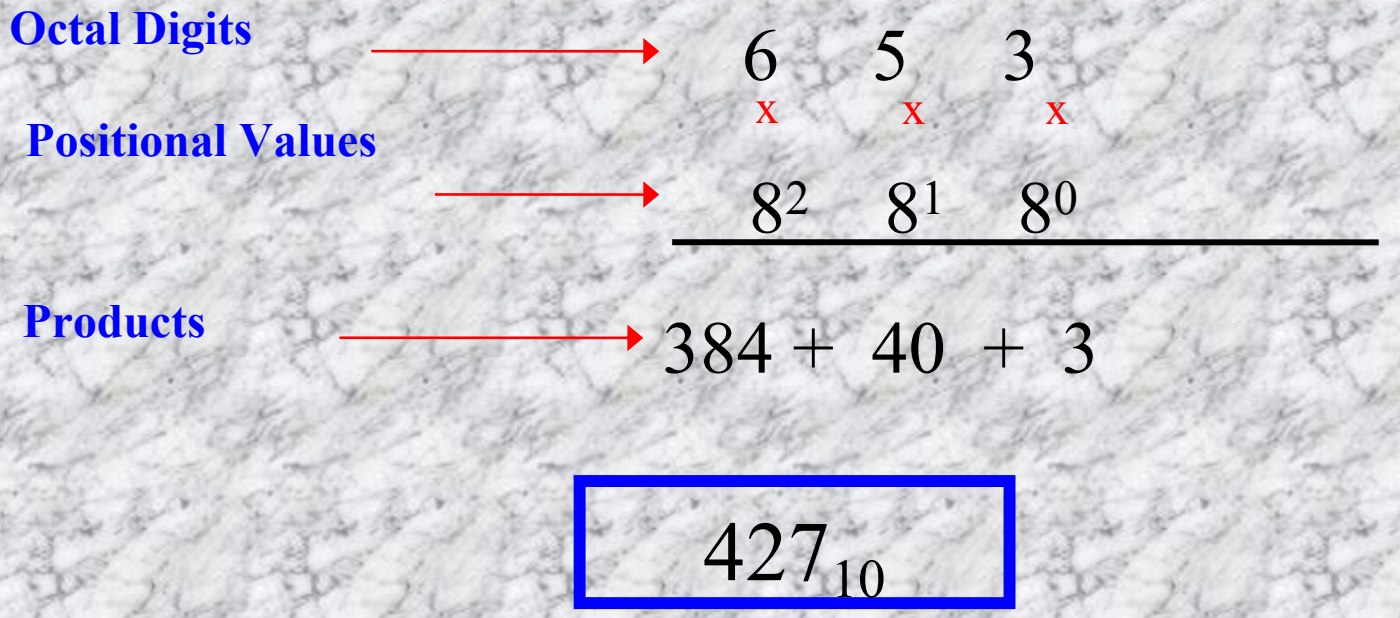
53 / 8 = 6 R= 5 Divide Q by 8

6 / 8 = 0 R= 6 Repeat until Q = 0



Octal to Decimal Conversion

Convert 653_8 to its decimal equivalent:



427_{10}

Octal to Binary Conversion

Each octal number converts to 3 binary digits

To convert 653_8 to binary, just substitute code:

Code
0 - 000
1 - 001
2 - 010
3 - 011
4 - 100
5 - 101
6 - 110
7 - 111

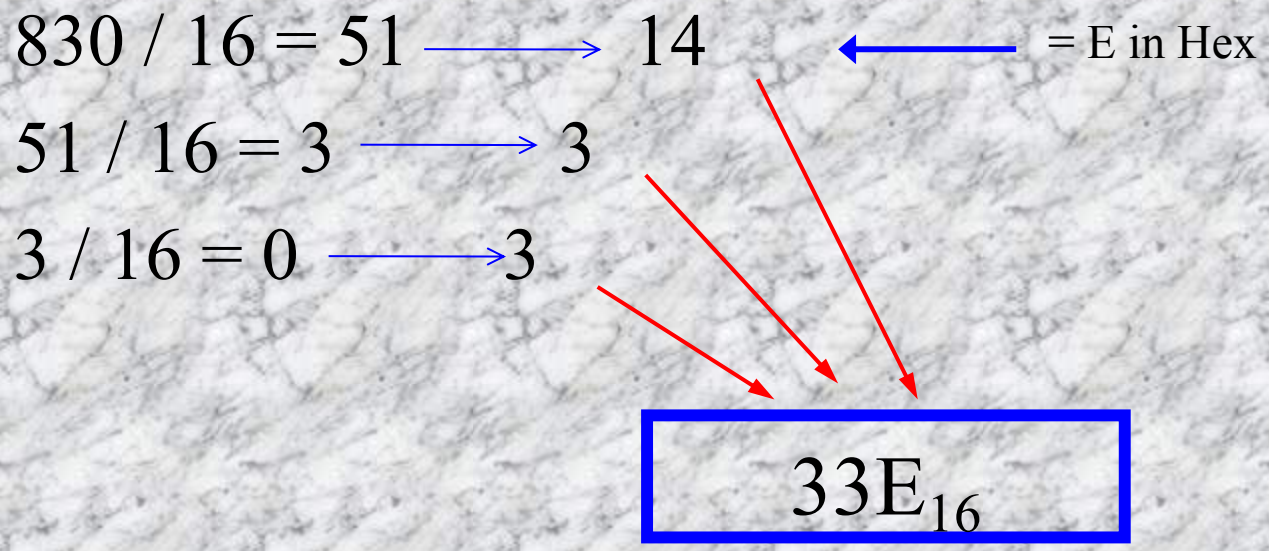
3. Hexadecimal Number System

- Base 16 system.
- Uses digits 0-9 & letters A,B,C,D,E,F.
- Groups of four bits represent each base 16 digit.

Decimal	Hexadecimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

Decimal to Hexadecimal Conversion

Convert 830_{10} to its hexadecimal equivalent:



Hexadecimal to Decimal Conversion

Convert 3B4F16 to its decimal equivalent:

Hex Digits → 3 B 4 F

Positional Values → 16^3 16^2 16^1 16^0

Products → 12288 +2816 +64 +15

$15{,}183_{10}$

Binary to Hexadecimal Conversion

- The easiest method for converting binary to hexadeci mal is to use a **substitution code**
- Each hex number converts to 4 binary digits

Substitution Code

0000 = 0	0100 = 4	1000 = 8	1100 = C
0001 = 1	0101 = 5	1001 = 9	1101 = D
0010 = 2	0110 = 6	1010 = A	1110 = E
0011 = 3	0111 = 7	1011 = B	1111 = F

Binary to Hexadecimal Conversion cont...

Convert **0101 0110 1010 1110 0110 1010_2** to hex using the 4-bit substitution code :

5	6	A	E	6	A
0101	0110	1010	1110	0110	1010

$56AE6A_{16}$

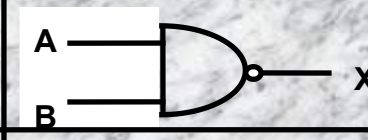
Binary to Octal Conversion

Substitution code can also be used to convert binary to octal by using 3-bit groupings:

2	5	5	2	7	1	5	2
010	101	101	010	111	001	101	010

$$25527152_8$$

Combinational Gates

Name	Symbol	Function	Truth Table
AND		X = A • B or X = AB	A B X 0 0 0 0 1 0 1 0 0 1 1 1
OR		X = A + B	A B X 0 0 0 0 1 1 1 0 1 1 1 1
Not		X = A`	A X 0 1 1 0
Buffer		X = A	A X 0 0 1 1
NAND		X = (AB)’	A B X 0 0 1 0 1 1 1 0 1 1 1 0
NOR		X = (A + B)’	A B X 0 0 1 0 1 0 1 0 0 1 1 0
XOR Exclusive OR		X = A ⊕ B or X = A’B + AB’	A B X 0 0 0 0 1 1 1 0 1 1 1 0
XNOR Exclusive NOR or Equivalence		X = (A ⊕ B)’ or X = A’B’+ AB	A B X 0 0 1 0 1 0 1 0 0 1 1 1

Assignment

From the book Solve:

- Section **1.3.7** (Pages23-24-25):
  - Points: **3,5,9,11,13,16,17,21**

- Section **1.4.2** (page29):
  - Points: **6,9**


Thank You