ME 168 (COMPUTER PROGRAMMING FOR ENGINEERS)

Dr. G.F.K. Tay

COLLEGE OF ENGINEERING
KNUST

Course Outline

- Course Objectives
  - To describe and explain some basic terminology used in computer programming
  - To write and run computer programs in MATLAB
  - To develop, compile and run code in C/C++
- Contact Hours
  - 2 hrs theory
  - 1 hr practicals

Contact Hours

- Kumasi Campus
  - MECH 1
    - Theory (Fridays, 8:00 – 10:00 am)
    - Lab (Mondays, 10:30 am – 3:00 pm)
  - AERO 1
    - Theory (Tuesdays, 8:00 – 10:00 am)
    - Lab (Tuesdays, 3:30 pm – 5:30 pm)
- Obuasi Campus (MECH 1)
  - Theory (Thursdays, 8:00 – 10:00 am)
  - Lab (Thursdays, 10:00 – 11:00 am)

Course Outline

- Course Content
  - Computer Programming Fundamentals
  - Computer Software
  - MATLAB Programming
  - C/C++, Object Oriented Programming

- Assessment
  - Quizzes, Assignments, Mid-Semester, Final Exams

Assessment

- Quizzes, Assignments ⟵ S1 = 30%
- Attendance ⟵
- Mid Semester Exams (S2 = 30%)
- End of Semester Exams (S3 = 70%)
- Final Score = (S1 + S2)/2 + S3

Class Regulations

- Be punctual (No lateness beyond 10 minutes)
- No use of mobile phones in class
- Decent conduct and we all ride into the sunset to live happily ever after (HEA)

References

- Chapra, S.C., *Applied Numerical Methods with MATLAB for Engineers and Scientists*, 3rd Int. Ed., 2012
- Nakamura, S., *Numerical Analysis and Graphic Visualization with MATLAB*, Prentice Hall Inc. 1996
- Any C/C++ Programming Book

Computer Programming

Relevant Terminology (Review)

- Computer Programming:

  The process of creating a computer program

- Computer Program:

  A set of instructions that can be executed by the computer to perform a specific task

- Computer Software

  A set of computer programs

How Do Computers Store Data?

- There are many types of data a computer needs to store or operate on, e.g.,
  - Text, numbers, images, sound, videos, programs, instructions, etc.,

- The data are first moved or loaded into memory before they are fetched by the CPU for processing

How Do Computers Store Data?

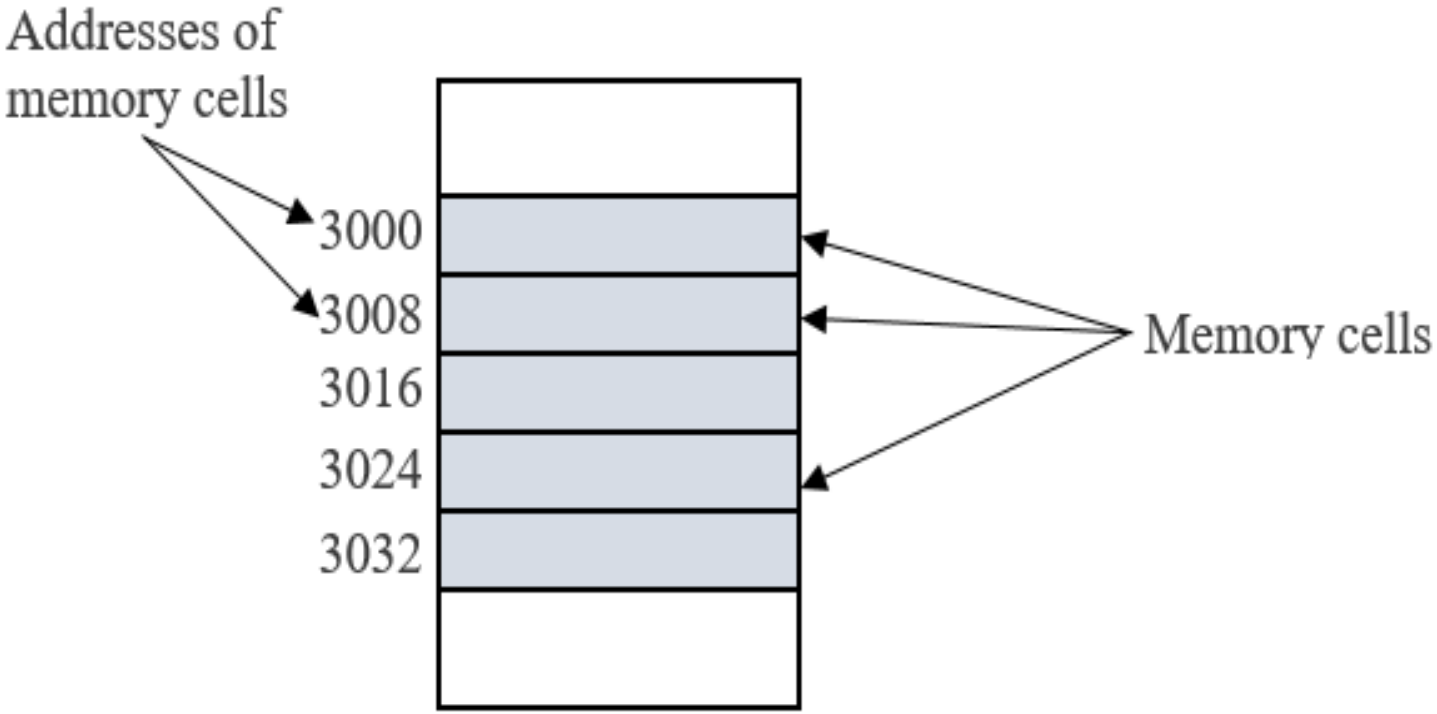
- The information/data in memory is stored as a series of switches (set ON/OFF)
- Each switch exists in two states: ON or OFF. Other representations are TRUE / FALSE or HIGH / LOW
  - If the switch is ON it's value is 1 (T or H)
  - If the switch is OFF it's value is 0 (F or L)
  - These two-state ON/OFF system 1 and 0 is called a binary number system

Bits, Nibbles and Bytes

- Each 0 or 1 in the binary system is called a **bit** (binary digit)
- A bit is the smallest possible unit of data a computer can recognize or use
- Bits are usually grouped together to store data, e.g.,
  - A group of 4 bits is called a **nibble**
  - 8 bits can be grouped together to form a **byte**
  - Other groupings are possible depending on register size (e.g., 32 bits or 64 bits)

Layout of Computer Memory

- A computer's memory consists of an ordered sequence of **bytes**
- Every byte in memory has a unique address



Storage of Numbers

- Numbers are stored as a series of bits
  - Example: How will the computer store the number 42?

- For a every grouping of bits, there is a fixed number of distinct numbers that can be represented in memory
  - This can be obtained by finding all of the distinct bit patterns that may be formed by the bits

Storage of Numbers

- How many different patterns with n bits?

Number of bits	Different Patterns
1	0, 1
2	00, 01, 10, 11
3	000, 001, 010, 011, 100, 101,110, 111

- Therefore, in general,
  - 1 bit yields 2 patterns
  - 2 bits yields 4 patterns
  - 3 bits yields 8 patterns

Storage of Numbers

- Mathematically,
  - 1 bit – 2 patterns
  - 2 bits – 4
  - 3 bits – 8
  - 4 bits – 16
  - 5 bits – 32
  - 6 bits – 64
  - 7 bits – 128
  - 8 bits – 256
  - n bits yields 2^n different patterns

Storage of Numbers

- One Byte – 256 Bit Patterns
  - E.g., 00000000, 00000001, 00000010, etc.
  - Converting the base two numbers to decimal yields values between 0 and 255
  - Thus, a byte can store numbers in the range 0 to 255
- How do you calculate the decimal values?
  - By number bases arithmetic

Binary to Decimal Conversion

Recall, any integral number may be written as

$$(d_{n-1}d_{n-2}\ldots d_2d_1d_0)_B$$

where *B* is the base or radix and d_i is the ith digit

- The decimal (base 10) equivalent of each digit is given by $d_i \times B^i$
- Thus, for *B* = 2, we have

$$(d_{n-1}d_{n-2}\ldots d_2d_1d_0)_2 = d_{n-1} \times 2^{n-1} + d_{n-2} \times 2^{n-2} + \cdots + d_2 \times 2^2 + d_1 \times 2^1 + d_0 \times 2^0$$

Binary to Decimal Conversion

$$(d_{n-1}d_{n-2}\ldots d_2d_1d_0)_2 = d_{n-1}\times 2^{n-1} + d_{n-2} \times 2^{n-2} + \cdots + d_2 \times 2^2 + d_1 \times 2^1 + d_0 \times 2^0$$

- By the above formula

  $00_2 = 0_{10}$; $01_2 = 1_{10}$;$10_2 = 2_{10}$; $11_2 = 3_{10}$

- This implies that 2 bits can be used to store or encode the numbers 0, 1, 2 and 3.
- 3 bits can be used to encode the numbers 0, 1, 2, 3, 4, 5, 6, 7.

Quiz 1

1) What are the different bit patterns than can be formed with n = 4 bits?
2) Evaluate the corresponding decimal numbers for your answers in question (1).
3) What is the largest number that can be encoded by n = 16 bits?

Bit Numbering (LSB and MSB)

Given the binary number $(d_{n-1}d_{n-2}\ldots d_2d_1d_0)_{two}$ with digits d_i the bits are numbered from right to as follows: 0, 1, 2, 3, …, n-2, n-1.

Example 1: The bits in the 8-bit number 11001011_2 are numbered as shown below

7	6	5	4	3	2	1	0
1	1	0	0	1	0	1	1

i.e., they are numbered from 0 to 7

Bit Numbering (LSB and MSB)

Similarly, the bits in the 16-bit number 0000000011001011_2 are numbered from 0 to 15 as shown below

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	1	1	0	1	0	0	1	1

- The rightmost bit (bit at the lowest position) is called the Least Significant Bit (LSB)
- The leftmost bit (bit at the highest position) is called the Most Significant Bit (MSB)

Least and Most Significant Byte

A multi-byte number or word

byte 3	byte 2	byte 1	byte 0

- The byte at the lowest position is called the Least Significant Byte (LSB)
- The byte at the highest position is called the Most Significant Byte (MSB)

MSB LSB

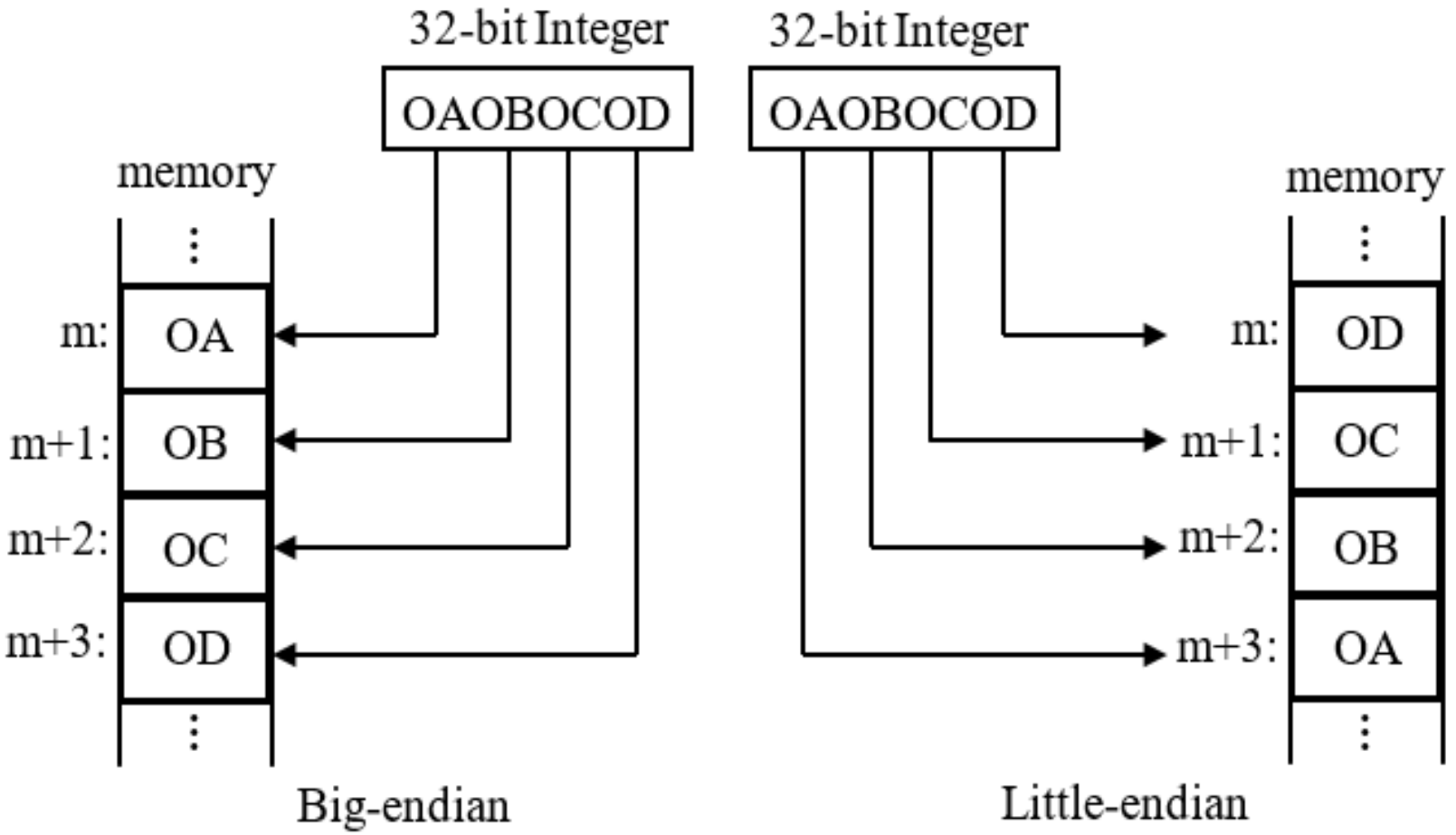
byte 3	byte 2	byte 1	byte 0

Least Significant Byte (LSB), Most Significant Byte (MSB) and Endianness

Computers differ in the way they store numbers

- Some computers store the MSB at the smallest memory address and the LSB at largest memory address. This is called Big-endian storage (e.g., Motorola Power PC, Sun Microsystems, etc)
- Other computers store the LSB at the smallest memory address and the MSB at the largest memory address. This is called Little-endian storage (e.g., Windows PC and MAC)

Big-endian vs Little-endian Storage



Representation of Signed (Positive and Negative) Numbers

Computer programs calculate both positive and negative numbers. Thus, it is necessary to distinguish between them.

- Signed numbers are represented using the Two's Complement Method
- Other methods such as the Signed Magnitude and 1's Complement have been used in the past, but found to be more cumbersome and less convenient compared to 2's complement

The 2's Complement Method

Uses leading zeros and ones to indicate sign

- Leading 0s indicate a positive number
- Leading 1s indicate a negative number

Recall, n = 3 bits can be used to store the numbers 0, 1, 2, 3, 4, 5, 6 and 7 only.

$$000_{two} = (0 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) = 0_{ten}$$
$$001_{two} = (0 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) = 1_{ten}$$
$$010_{two} = (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) = 2_{ten}$$
$$011_{two} = (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) = 3_{ten}$$
$$100_{two} = (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) = 4_{ten}$$
$$101_{two} = (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) = 5_{ten}$$
$$110_{two} = (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) = 6_{ten}$$
$$111_{two} = (1 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) = 7_{ten}$$

The 2's Complement Method

If negative numbers are to be included, the 3 bits can be used to store the following positive and negative numbers:

$$000_{two} = (0 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) = 0_{ten}$$
$$001_{two} = (0 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) = 1_{ten}$$
$$010_{two} = (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) = 2_{ten}$$
$$011_{two} = (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) = 3_{ten}$$
$$100_{two} = (1 \times -2^2) + (0 \times 2^1) + (0 \times 2^0) = -4_{ten}$$
$$101_{two} = (1 \times -2^2) + (0 \times 2^1) + (1 \times 2^0) = -3_{ten}$$
$$110_{two} = (1 \times -2^2) + (1 \times 2^1) + (0 \times 2^0) = -2_{ten}$$
$$111_{two} = (1 \times -2^2) + (1 \times 2^1) + (1 \times 2^0) = -1_{ten}$$

The 2's Complement Method

$$000_{two} = (0 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) = 0_{ten}$$
$$001_{two} = (0 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) = 1_{ten}$$
$$010_{two} = (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) = 2_{ten}$$
$$011_{two} = (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) = 3_{ten}$$
$$100_{two} = (1 \times -2^2) + (0 \times 2^1) + (0 \times 2^0) = -4_{ten}$$
$$101_{two} = (1 \times -2^2) + (0 \times 2^1) + (1 \times 2^0) = -3_{ten}$$
$$110_{two} = (1 \times -2^2) + (1 \times 2^1) + (0 \times 2^0) = -2_{ten}$$
$$111_{two} = (1 \times -2^2) + (1 \times 2^1) + (1 \times 2^0) = -1_{ten}$$

That is, the 3 bits can be used to encode the following signed numbers: -4, -3, -2, -1, 0, 1, 2, 3

Representation of Signed Numbers

Self-assessment: By means of calculations determine the set of signed numbers that can be encoded by exactly

n = 4 bits.

- In computer programming, the sets of signed and unsigned numbers are called Fundamental Numeric Data Types.
- All programming languages such as MATLAB and C/C++ define and use fundamental data types

Some Fundamental Data Types

C/C++ Type	MATLAB Version	Size in Bytes	Range of Values
short	int16	2	-32,768 to 32,767
unsigned short	uint16	2	0 to 65,535
int	int32	4	-2,147,483,648 to 2,147,483,647
unsigned int	uint32	4	0 to 4,294,967,295
long	–	4	-2,147,483,648 to 2,147,483,647
unsigned long	–	4	0 to 4,294,967,295
long long	int64	8	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
unsigned long long	uint64	8	0 to 18,446,744,073,709,551,615
float	single	4	-3.4×10^{38} to 3.4×10^{38} with approximately 7-digits accuracy
double	double	8	-1.7×10^{308} to 1.7×10^{308} with approximately 15-digits accuracy
long double	-	10 or 16	At least as much precision as double

Character Encoding (ASCII and Unicode)

Character encoding is the conversion text data into binary numbers.

To store text data in memory each character is first converted into a unique decimal number using some encoding system.

- The most widely used encoding system is ASCII
- ASCII stands for American Standard Code for Information Interchange
  - Developed in the US, hence encodes only English characters, numbers and other symbols

ASCII Encoding Scheme

The ASCII encoding scheme has a character set containing a total of 128 characters

- Each character is assigned a unique value between 0 and 127
- Question: How many bits are required to store a number between 0 and 127?
  - *Answer* = 7 bits: Since only 7 bits are required for storage, ASCII characters by convention are stored with 8 bits (= 1 byte)
  - In the 8 bits, the most significant bit (MSB) is set to 0

ASCII Character Codes

Character	ASCII Code	Binary Value
Space	32	0010 0000
!	33	0010 0001
“	34	0010 0010
#	35	0010 0011
$	36	0010 0100
%	37	0010 0101
&	38	0010 0110
‘	39	0010 0111
(	40	0010 1000
)	41	
*	42	
+	43	
,	44	

ASCII Character Codes

Character	ASCII Code	Binary Value
-	45	
.	46	
/	47	
0	48	
1	49	
2	50	
3	51	
4	52	
5	53	
6	54	
7	55	
8	56	
9	57	

ASCII Character Codes

Character	ASCII Code	Binary Value
:	58	
;	59	
<	60	
=	61	
>	62	
?	63	
@	64	
A	65	
B	66	
C	67	
D	68	
E	69	
F	70	

ASCII Character Codes

Character	ASCII Code	Binary Value
G	71	
H	72	
I	73	
J	74	
K	75	
L	76	
M	77	
N	78	
O	79	
P	80	
Q	81	
R	82	
S	83	

ASCII Character Codes

Character	ASCII Code	Binary Value
T	84	
U	85	
V	86	
W	87	
X	88	
Y	89	
Z	90	
[	91	
\	92	
]	93	
^	94	
_	95	
`	96	

ASCII Character Codes

Character	ASCII Code	Binary Value
a	97	
b	98	
c	99	
d	100	
e	101	
f	102	
g	103	
h	104	
i	105	
j	106	
k	107	
l	108	
m	109	

ASCII Character Codes

Character	ASCII Code	Binary Value
n	110	
o	111	
p	112	
q	113	
r	114	
s	115	
t	116	
u	117	
v	118	
w	119	
x	120	
y	121	
z	122	

ASCII Character Codes

Character	ASCII Code	Binary Value
{	123	
|	124	
}	125	
~	126	
Delete	127	

Limitation: The 1 byte or 8 bits of storage can be used to represent only 128 unique characters. This is not enough to represent all the world's characters.

- Therefore more numbers will be needed to cater for all characters in the world, including Chinese, Japanese, Korean and Greek symbols, among others

Unicode Encoding Scheme

- Solves ASCII's problem of using a small set of numbers
- Uses 2 bytes (16 bits) instead of 1 byte to store a character
- The 16 bits corresponds to a set of 2^{16} = 65536 different numbers, i.e., numbers ranging from 0 to 65535
  - Enough to encode nearly all of the world's characters, including Chinese, Japanese, Greek symbols, etc.

Hexadecimal Number System

- Hexadecimal numbers are numbers in base 16
  - Provides a more compact notation than the binary system
  - More convenient when writing large numbers
  - Therefore, is the more preferred number system for programming languages

- It has 16 digits as follows: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Hexadecimal Number System

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A or a
11	1011	B or b
12	1100	C or c

Hexadecimal Number System

Decimal	Binary	Hexadecimal
13	1101	D or d
14	1110	E or e
15	1111	F or f

- Each hex digit represents 4 binary bits
  - e.g., $FFF0_{hex} = 1111\ 1111\ 1111\ 0000_{two}$

- Convert $0001\ 0011\ 0101\ 0111\ 1011_{two}$ to a hexadecimal number