Microprocessor Systems and Interfacing

EEE 342

Nesruminallah

nesruminallah@cuilahore.edu.pk

About the Instructor

- MSc, Communication Engineering, 2014
  - University of Portsmouth, United Kingdom, .
- BSc, Telecommunication Engineering, 2012
  - UET Peshawar , Pakistan.

Course Introduction

- Textbook
  - The Intel Microprocessors by Barry B. Brey, 8th Edition, Pearson
  - Assembly Language Programming and Organization of the IBM PC by Ytha Yu and Charles Marut, 1992, 1st Edition, McGraw-Hill
  - Embedded Systems: Introduction to Arm® Cortex™-M Microcontrollers, by Jonathan W Valvano, Vol 1, 5th Edition, 2019, CreateSpace

Grading Policy

- Assignments 10%
  - Minimum 4
- Quizzes (scheduled/surprised) 15%
  - Minimum 4
- Midterm 25%
- Final exam 50%

Academic Honesty

- Your work and participation in the course **must** be your own
- If students are found to have collaborated excessively or to have cheated (e.g. by copying or sharing answers in assignments or during an examination), all involved will at a minimum receive grades of 0 for the first infraction
- Further infractions may result in failure in the course

Lectures

- Lecture notes given by instructor.

- Please be courteous in class
  - Arrive on time
  - Turn off cell phones / sound on laptop
  - Keep quiet …
  - Drinking or eating is strictly prohibited

- Attendance is important
  - There are just things that you cannot learn from reading notes
  - 80% is must to appear in final exam and pass the course

Few Recommendations

- “Eighty percent of success is showing up.”
  - Come to lectures, discussions, lab
- If you are not sure: ask!
  - Talk to us after class, send email, come to office hours
  - Early communication solves problems easiest
  - Don’t wait until it’s too late
- Email protocol
  - Write your full name and registration ID.
    - We need to know who you are.
  - Do not forget to write subject of email
  - Be professional

Couse CLOs and PLOs

- **Theory CLOs**:
- CLO-1: To write the Intel-assembly code using the knowledge of programmer model, addressing mode and assembly language programming concepts. (PLO3-C5)
- CLO-2: To integrate the memory, timer, I/O and PPI with microprocessor using address decoding techniques. (PLO3-C5)
- CLO-3: To design digital system based on microprocessor using the knowledge of architecture, memory, timer, I/O and PPI interfacing. (PLO3-C5)
- **Lab CLOs**:
- CLO4: To explain and reproduce the Intel-assembly and STM32F407VG C-Programming codes using software and hardware platforms. (PLO5-P3)
- CLO5: To design digital system using the knowledge of STM32F407VG C-Programming and peripherals. (PLO3-C5)
- CLO6: To write effective report(s) of the assigned project. (PLO10-A2).
- CLO7: To describe the impact of digital system on our society and environment using existing industrial standards (PLO7-C6).
- CLO8: To justify the significance of designed project to the society using existing engineering practices (PLO6-C6).

Course Contents

- Introduction to microprocessor and microcontroller and
- Basic concepts and definitions of computer architecture and organizations
- Introduction to Programmers model of 8086/88
- Assembly Language Programming for 8086/88 Architecture
- Interfacing of RAM/ROM with 8088 microprocessors
- Introduction to Microcontroller (STM32F407VG)
- Interfacing of RAM/ROM with 8086 microprocessors
- Stack programming and memory mapping
- 8254 timer/counter interfacing with 8088 microprocessors
- I/O interfacing (isolated and memory-mapped) with 8088 microprocessors
- 8255 PPI interfacing with 8088 microprocessors
- A/D and D/A Conversion
- Hardware Interrupts
- Interfacing output devices with 8088 microprocessors using PPI

Number System and Conversions

CLO	Bloom Taxonomy	Specific Outcome
CLO1	C2	Comprehend the theoretical knowledge of number systems such as Binary, Octal, Decimal and Hexadecimal numbers using standard conversion methods.

- Outline
  - Binary numbers
  - Number-Base conversions
  - Octal and hexadecimal numbers
  - Complements and signed binary numbers

Digital Computer Systems

- Digital systems consider *discrete* amounts of data.

- Examples
  - 26 letters in the alphabet
  - 10 decimal digits

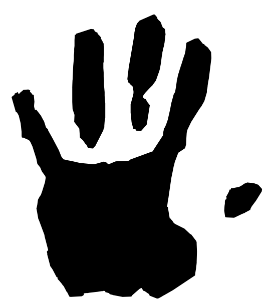
- Larger quantities can be built from discrete values
  - Words made of letters
  - Numbers made of decimal digits (e.g. 239875.32)

Digital Computer Systems

- Questions to ask
  - How the numbers are represented in digital systems?
  - How computer performs basic arithmetic operations?
- Computers operate on *binary* values (0 and 1)
- Easy to represent binary values electrically
  - Voltages and currents
  - Advantages
    - Can be implemented using circuits
    - Create the building blocks of modern computers

Understanding Decimal Numbers

- Decimal numbers are made of decimal digits: (0,1,2,3,4,5,6,7,8,9)
- But how many items does a decimal number represent?
  - $8653 = 8x10^3 + 6x10^2 + 5x10^1 + 3x10^0$
- What about fractions?
  - $97654.35 = 9x10^4 + 7x10^3 + 6x10^2 + 5x10^1 + 4x10^0 + 3x10^{-1} + 5x10^{-2}$
  - In formal notation -> $(97654.35)_{10}$
- Why do we use 10 digits, anyway?



Understanding Binary Numbers

- Binary numbers are made of binary digits (bits)
  - 0 and 1
- How many items does a binary number represent?
  - $(1011)_2 = 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 = (11)_{10}$
- What about fractions?
  - $(110.10)_2 = 1x2^2 + 1x2^1 + 0x2^0 + 1x2^{-1} + 0x2^{-2}$
- Groups of eight bits are called a *byte*
  - $(11001001)_2$
- Groups of four bits are called a *nibble*
  - $(1101)_2$

Understanding Octal Numbers

- Octal numbers are made of octal digits
  - 0,1,2,3,4,5,6,7
- How many items does an octal number represent?
  - $(4536)_8 = 4x8^3 + 5x8^2 + 3x8^1 + 6x8^0 = (1362)_{10}$
- What about fractions?
  - $(465.27)_8 = 4x8^2 + 6x8^1 + 5x8^0 + 2x8^{-1} + 7x8^{-2}$
- Octal numbers don’t use digits 8 or 9
- Why would someone use octal number, anyway?

Understanding Hexadecimal Numbers

- Hexadecimal numbers are made of 16 digits
  - (0,1,2,3,4,5,6,7,8,9,A, B, C, D, E, F)
- How many items does an hex number represent?
  - $(3A9F)_{16} = 3x16^3 + 10x16^2 + 9x16^1 + 15x16^0 = 14999_{10}$
- What about fractions?
  - $(2D3.5)_{16} = 2x16^2 + 13x16^1 + 3x16^0 + 5x16^{-1} = 723.3125_{10}$
- Note that *each* hexadecimal digit can be represented with four bits.
  - $(1110)_2 = (E)_{16}$

Exercise (Convert to decimal)

- $(1011.101)_2$
- Answer = 11.625

- $(24.6)_8$
- Answer = 20.75

- $(IBC2)_{16}$
- Answer = 7106

Putting It All Together

Decimal	Binary	Octal	Hexadecimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Convert an Integer from Decimal to Another Base

- For each digit position
  - Divide decimal number by the base (e.g. 2)
  - The *remainder* is the lowest-order digit
  - Repeat first two steps until no *divisor* remains

Example for $(13)_{10}$

	Integer Quotient		Remainder	Coefficient
13/2 =	6	+	½	$a_0 = 1$
6/2 =	3	+	0	$a_1 = 0$
3/2 =	1	+	½	$a_2 = 1$
1/2 =	0	+	½	$a_3 = 1$

Answer $(13)_{10} = (a_3\, a_2\, a_1\, a_0)_2 = (1101)_2$

Exercise (Convert decimal to other bases)

- 291 to binary
- Answer = $(100100011)_2$

- 291 to octal
- Answer = $(443)_8$

- 291 to hexadecimal
- Answer = $(123)_{16}$

Convert a Fraction from Decimal to Another Base

- For each digit position
  - Multiply decimal number by the base (e.g. 2)
  - The integer is the highest-order digit
  - Repeat first two steps until fraction becomes zero

Example for $(0.625)_{10}$

	Integer		Fraction	Coefficient
$0.625 \times 2 =$	1	+	0.25	$a_{-1} = 1$
$0.250 \times 2 =$	0	+	0.50	$a_{-2} = 0$
$0.500 \times 2 =$	1	+	0	$a_{-3} = 1$

Answer $(0.625)_{10} = (0.a_{-1}\, a_{-2}\, a_{-3})_2 = (0.101)_2$

Exercise (Convert fraction to decimal)

- $(0.513)_{10}$ to octal
- Answer = $(0.406517\ldots)_8$

- 0.513 x 8 = 4.104
- 0.104 x 8 = 0.832
- 0.832 x 8 = 6.656
- 0.656 x 8 = 5.248
- 0.248 x 8 = 1.984
- 0.984 x 8 = 7.872

Binary Addition

- Binary addition is very simple.
- An example of adding two binary numbers

```
    1   1   1   1   1   1 <--- carries

        1   1   1   1   0   1
+           1   0   1   1   1
-----------------------------
    1   0   1   0   1   0   0
```

Binary Subtraction

- We can also perform subtraction (with borrows in place of carries)
- Example: subtract $(10111)_2$ from $(1001101)_2$

$$
\begin{array}{cccccccc}
 & & 1 & & & 10 & & \leftarrow \texttt{borrows} \\
 & 0 & \cancel{10} & 10 & 0 & \cancel{0} & 10 & \\
 & \cancel{1} & \cancel{0} & \cancel{0} & \cancel{1} & \cancel{1} & \cancel{0} & 1 \\
- & & & 1 & 0 & 1 & 1 & 1 \\
\hline
 & & 1 & 1 & 0 & 1 & 1 & 0
\end{array}
$$

Binary Multiplication

- Binary multiplication is much the same as decimal multiplication
  - The multiplication operations are much simpler

```
               1  0  1  1  1
X                 1  0  1  0
-----------------------------
               0  0  0  0  0
            1  0  1  1  1
         0  0  0  0  0
      1  0  1  1  1
-----------------------------
      1  1  1  0  0  1  1  0
```

Converting Between Base 16 and Base 2

$$3A9F_{16} = \underbrace{0011}_{3}\ \underbrace{1010}_{A}\ \underbrace{1001}_{9}\ \underbrace{1111}_{F}{}_2$$

- Conversion is easy
  - Determine 4-bit value for each hex digit
- Note that there are $2^4 = 16$ different values of four bits
- Easier to read and write in hexadecimal
- Representations are equivalent

Converting Between Base 16 and Base 8

$$3A9F_{16} = \underbrace{0011}_{3}\ \underbrace{1010}_{A}\ \underbrace{1001}_{9}\ \underbrace{1111_2}_{F}$$

$$35237_8 = \underbrace{011}_{3}\ \underbrace{101}_{5}\ \underbrace{010}_{2}\ \underbrace{011}_{3}\ \underbrace{111_2}_{7}$$

1. Convert from Base 16 to Base 2
2. Regroup bits into groups of three starting from right
3. Ignore leading zeros
4. Each group of three bits forms an octal digit

Exercise

- Convert base 16 to base 8 without intermediate stage of base 10

- $(B98D)_{16}$

- Answer = $(134615)_8$

Converting Between Base 8 and Base 16

- $(673.124)_8 = (110\ \ 111\ \ 011\ .\ 001\ \ 010\ \ 100\ )_2$

$$= (\underline{1}\ \ \underline{1011}\ \ \underline{1011}\ .\ \ \underline{0010}\ \ \underline{1010}\ \ 0\ )$$

$$= (\underline{1}\ \ \underline{B}\ \ \underline{B}\ .\ \ \underline{2}\ \ \underline{A})$$

Complements

- Used in computers to simplify the subtraction operation
- For each base-r system
  - Diminished radix complement or r-1's complement
  - Radix complement or r's complement
- For example for base-2 system
  - 1's complement
  - 2's complement
- For base-10 system
  - 9's complement
  - 10's complement

Diminished Radix Complement (r-1's complement)

- Given a number 'N' in base 'r' with 'n' digits, r-1's complement is defined as $(r^n-1) – N$
- For example if $N = (546700)_{10}$ then r = 10 and n = 6
  - $(r^n-1) = 10^6 – 1 = 999999$
  - 9's complement of $N = (r^n-1) – N = 999999 – 546700 = 453299$
  - Similarly for $N = (012398)_{10}$, 9's complement of N is $999999 – 012398 = 987601$

Diminished Radix Complement (r-1's complement)

- For a binary number, r = 2 and r-1 or 1's complement can be found just like base-10 numbers
- For example if N = $(1010)_2$ then r = 2 and n = 4
  - $(r^n-1) = 2^4 - 1 = (15)_{10} = (1111)_2$
  - 1's complement of N = $(r^n-1) - N = 1111 - 1010 = (0101)_2$
- Shortcut: Invert all the bits of N in order to take its 1's complement
  - 1's complement of 1011000 → 0100111
  - 1's complement of 0101101 → 1010010

Radix Complement (r's complement)

- Given a number 'N' in base 'r' with 'n' digits, r's complement is defined as $r^n - N$ for $N \neq 0$ and 0 for $N = 0$
  - r's complement can also be obtained by adding 1 to r-1's complement
- For example if $N = (546700)_{10}$
  - 9's complement of $N = (r^n-1) - N = 999999 - 546700 = 453299$
  - 10's complement of N = 9's complement + 1 = $r^n - N$ = 453300

Radix Complement (r's complement)

- For a binary number, r = 2 and r or 2's complement can be found just like base-10 numbers
- For example if N = $(1010)_2$ then r = 2 and n = 4
  - 1's complement of N = (r^n-1) – N = 1111 – 1010 = $(0101)_2$
  - 2's complement of N = 1's complement of N + 1 = 0110

2's Complement Shortcuts

- **Algorithm 1**
  - Complement each bit and then add 1 to the result
  - Example: Find the 2's complement of $(01100101)_2$ and of its 2's complement

N = 01100101	[N] = 10011011
10011010	01100100
+ 1	+ 1
---------------	---------------
10011011	01100101

2's Complement Shortcuts

- **Algorithm 2**
  - Starting with the least significant bit, copy all of the bits up to and including the first 1 bit and then complementing the remaining bits
  - Example: Find the 2's complement of $(01100101)_2$

N = 0 1 1 0 0 1 0 1

[N] = 1 0 0 1 1 0 1 1

Signed Numbers and their representation

- Plus and minus sign used for decimal numbers: 25 (or +25), -16, etc.
- For computers, desirable to represent everything as *bits*
- Three types of signed binary number representations
  - signed magnitude, 1's complement, 2's complement
- In each case: left-most bit indicates sign: positive (0) or negative (1)

<u>signed magnitude</u>

$0\underline{0001100}_2 = 12_{10}$

Sign bit Magnitude

$1\underline{0001100}_2 = -12_{10}$

Sign bit Magnitude

1's Complement Representation

- Invert all bits
  - 00110011 → 11001100
  - 10101010 → 01010101
- For an n bit number N, the 1's complement is $(2^n - 1) - N$
- To find negative of 1's complement number take the 1's complement

$00001100_2 = 12_{10}$

Sign bit Magnitude

$11110011_2 = -12_{10}$

Sign bit Magnitude

2's Complement Representation

- Invert all bits and add 1
  - 00110011 → 11001101
  - 10101010 → 01010110
- For an n bit number N the 2's complement is $(2^n-1) – N + 1$
- To find negative of 2's complement number take the 2's complement

$00001100_2 = 12_{10}$

Sign bit Magnitude

$11110100_2 = -12_{10}$

Sign bit Magnitude

1's Complement Addition

- Add $+(1100)_2$ and $+(0001)_2$
  - $(12)_{10} = +(1100)_2 = 01100_2$
  - $(1)_{10} = +(0001)_2 = 00001_2$

			0	1	1	0	0
Add	+		0	0	0	0	1
		0	0	1	1	0	1
Add carry		└→					0
Final Result			0	1	1	0	1

Step 1: Add binary numbers
Step 2: Add carry to low-order bit

1's Complement Subtraction

- Subtract $+(0001)_2$ from $+(1100)_2$
  - $(12)_{10} = +(1100)_2 = 01100_2$
  - $(-1)_{10} = -(0001)_2 = 11110_2$

```
                 0 1 1 0 0
          −      0 0 0 0 1
          -----------------
1's comp
                 0 1 1 0 0
Add       +      1 1 1 1 0
          -----------------
               1 0 1 0 1 0
Add carry                1
          -----------------
Final Result     0 1 0 1 1
```

Step 1: Take 1's complement of 2nd operand
Step 2: Add binary numbers
Step 3: Add carry to low order bit

2's Complement Addition

- Add $+(1100)_2$ and $+(0001)_2$.
  - $(12)_{10} = +(1100)_2 = 01100_2$
  - $(1)_{10} = +(0001)_2 = 00001_2$

$$
\begin{array}{lcccccc}
 & & 0 & 1 & 1 & 0 & 0 \\
\text{Add} & + & 0 & 0 & 0 & 0 & 1 \\
\hline
\text{Final Result} & 0 & \boxed{0} & \boxed{1} & \boxed{1} & \boxed{0} & \boxed{1}
\end{array}
$$

Ignore

Step 1: Add binary numbers
Step 2: Ignore carry bit

2's Complement Subtraction

- Subtract $+(0001)_2$ from $+(1100)_2$
  - $(12)_{10} = +(1100)_2 = 01100_2$
  - $(-1)_{10} = -(0001)_2 = 11111_2$

```
      0 1 1 0 0
  -   0 0 0 0 1
  -------------
2's comp

      0 1 1 0 0
Add + 1 1 1 1 1
  -------------
Final Result
    1 0 1 0 1 1

Ignore Carry
```

Step 1: Take 2's complement of 2nd operand
Step 2: Add binary numbers
Step 3: Ignore carry bit

2's Complement Subtraction: Example 2

- Let's compute $(13)_{10} - (5)_{10}$
  - $(13)_{10} = +(1101)_2 = (01101)_2$
  - $(-5)_{10} = -(0101)_2 = (11011)_2$
- Adding these two 5-bit codes

```
                        0 1 1 0 1
carry             +     1 1 0 1 1
                  ---------------
                    1   0 1 0 0 0
```

- Discarding the carry bit, the sign bit is seen to be zero, indicating a correct result. Indeed,

  $(01000)_2 = +(1000)_2 = +(8)_{10}$

2's Complement Subtraction: Example 3

- Let's compute $(5)_{10} - (12)_{10}$
  - $(-12)_{10} = -(1100)_2 = (10100)_2$
  - $(5)_{10} = +(0101)_2 = (00101)_2$
- Adding these two 5-bit codes

```
        0 0 1 0 1
+       1 0 1 0 0
--------------
        1 1 0 0 1
```

- Here, there is no carry bit and the sign bit is 1
  - This indicates a negative result, which is what we expect. $(11001)_2 = -(7)_{10}$

Subtraction using r’s complements

- Subtraction of two n-digit unsigned numbers M-N in base r is performed as follows
  - Add M to r’s complement of N
  - If $M \geq N$ sum will produce an end carry which can be discarded
  - If $M < N$ sum does not produce end carry. Take r’s complement of the result to know exact result

Subtraction using r's complements

- Let $M = (52532)_{10}$, $N = (3250)_{10}$, $M - N = ?$

  M = 52532

  N = 03250

  10's complement of N = 96750

  M – N = 52532
  +96750

  149282

  Discard end carry and the result is

  Answer = $(49282)_{10}$

Subtraction using r's complements

- Let M = $(3250)_{10}$, N = $(72532)_{10,}$ M – N = ?

M =	03250
N =	72532
10's complement of N =	27468
M – N =	03250
	+27468
	30718

No end carry in this case

Answer = - (10's complement of M - N) = - 69282

Same rules can be applied to base-2 numbers

Subtraction using r-1's complements

- Subtraction of two n-digit unsigned numbers M-N in base r is performed as follows
  - Add M to r-1's complement of N
  - If $M \geq N$ sum will produce an end carry which will be added to least significant digit of the sum
  - If $M < N$ sum does not produce end carry. Take r-1's complement of the result to know exact answer

Subtraction using r-1's complements

- Let M = $(1010100)_2$, N = $(1000011)_2$, M – N = ?

M =	1010100
N =	1000011
1's complement of N =	0111100
M – N =	1010100
	+0111100
	10010000
Add end carry around	
Answer =	$(0010001)_2$

Subtraction using r-1's complements

- Let M = $(1000011)_2$, N = $(1010100)_{2,}$ M – N = ?

  M = 1000011

  N = 1010100

  1's complement of N = 0101011

  M – N = 1000011

  +0101011

  1101110

  No end carry in this case

  Answer = - (1's complement of M - N) = -(0010001)