Chapter 1
Introduction to Microcontrollers

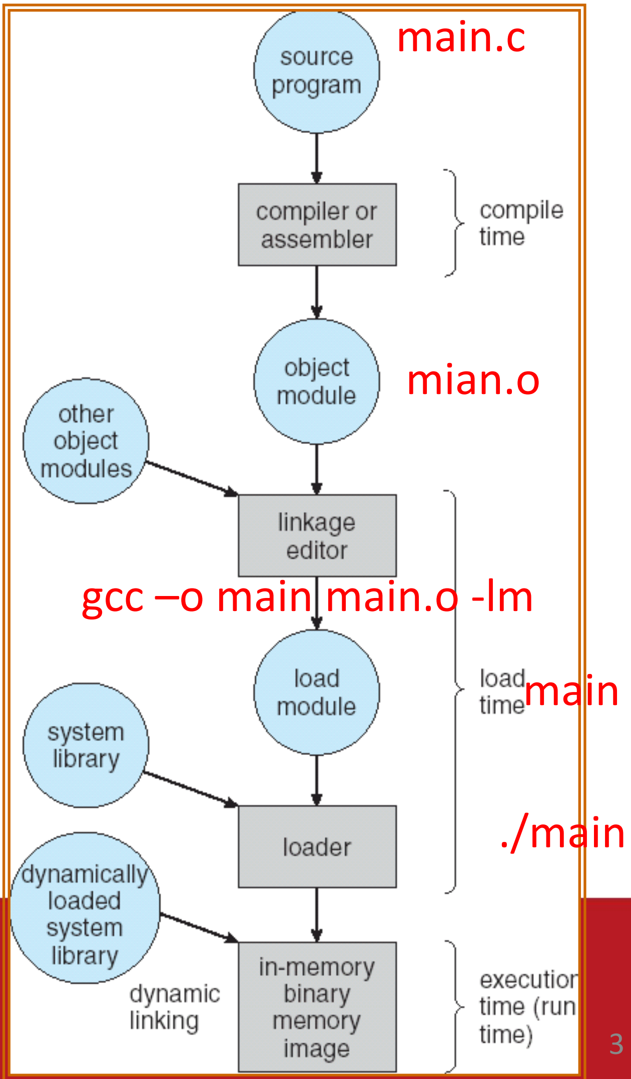
Chung-Ping Young (楊中平)

What is a Computer?

- A computer is made up of **hardware** and software.
    - **Central processing unit (CPU)**: performing computational operations, coordination of resources
    - **Input devices**: used to enter the program to be executed and data to be processed
    - **Output devices**: for display or print
    - **Memory**: programs/data must be stored in memory devices so that the processor can readily access them

What is a Computer?

- A computer is made up of hardware and **software**.
  - Computer software consists of a collection of programs.
  - Programs contain instructions and data for performing a specific task.
  - All programs must be translated into binary prior to execution by a compiler or an assembler.



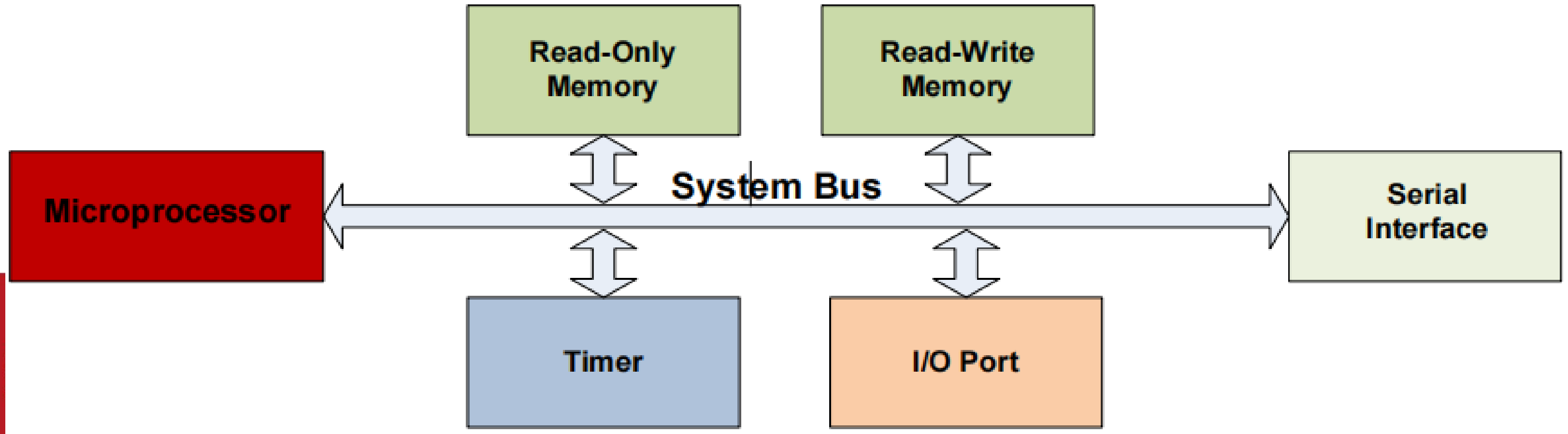
National Cheng Kung University

Assembly Language Program

- An instruction may include *immediate data* or the *address of data*.
- During instruction execution, the computer obtains immediate data from the "*program memory*" and obtain data with address from the "*data memory*".

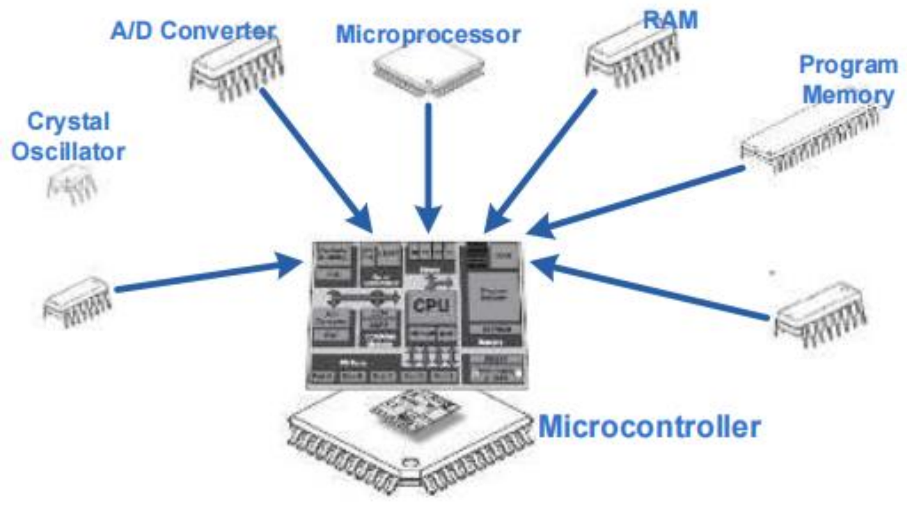
Microcomputer

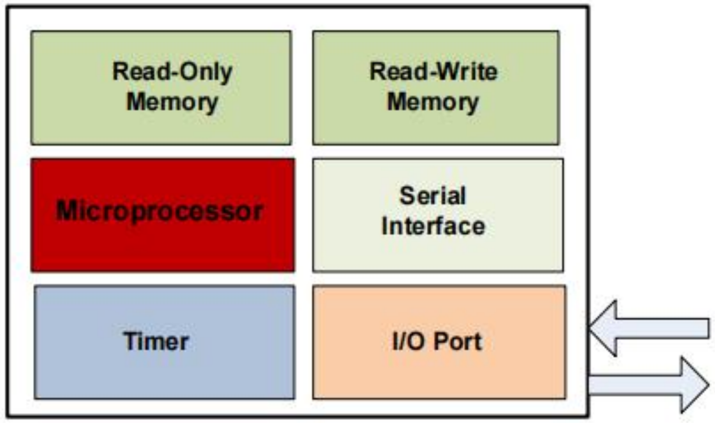
- Due to advances in semiconductor technology, it is possible to fabricate a CPU on a single chip. The result is a *microprocessor*.
- Along with the microprocessor chip, appropriate memory and I/O chips can be used to design a *microcomputer*.
- Single-chip microcomputers such as the Intel 8048 were used in a wide range of industrial and home applications.



Microcontroller

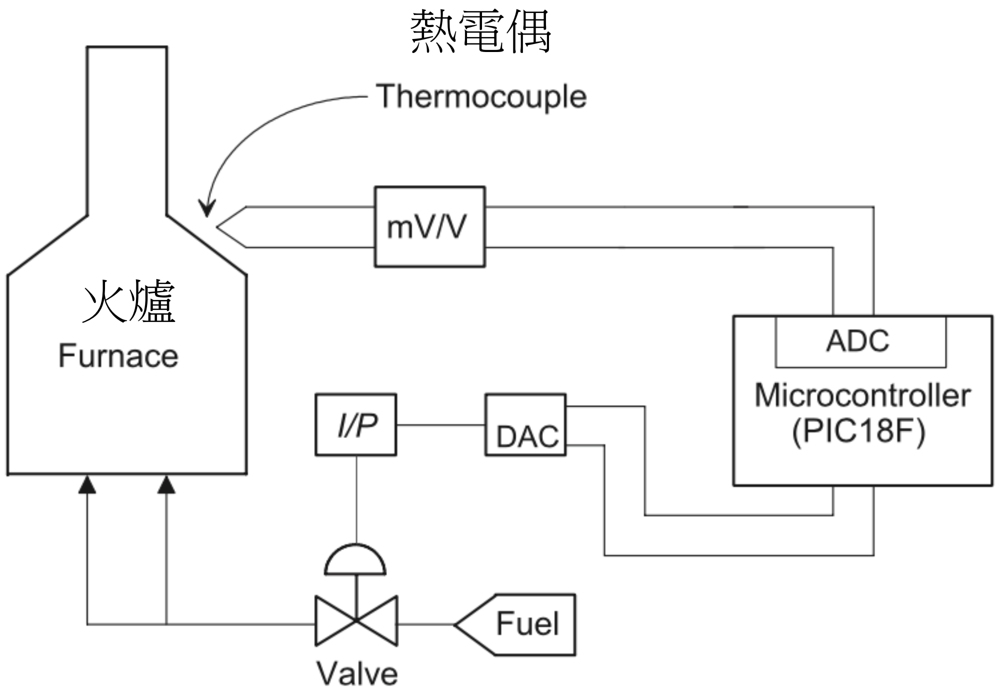
- *Microcontrollers* evolved from single-chip microcomputers. Microcontrollers are normally used for *dedicated applications* such as automotive systems, home appliances, and home entertainment systems.
- Typical microcontrollers include a CPU, memory, I/O, along with certain peripheral functions such as timers, and ADC (Analog-to-Digital Converter) all in a single chip.





Microcontroller

- A microcontroller-based temperature control system



Explanation of Terms

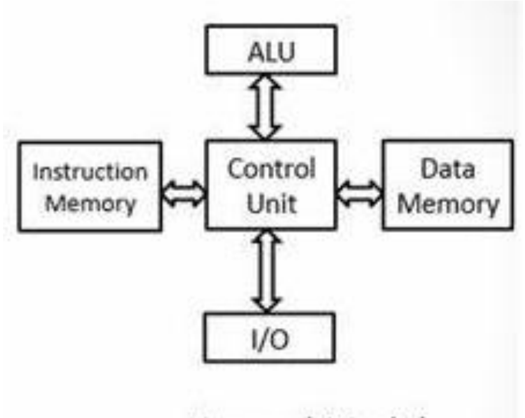
- *Address* is a pattern of 0's and 1's that represents a specific location in memory or a particular I/O device.
- *Arithmetic-logic unit* (ALU) is a digital circuit that performs arithmetic and logic operations on **two *n*-bit numbers**. The value of ***n*** for microcontrollers can be **8-bit or 16-bit or 32-bit**.
    - Microship's PIC18F contains an 8-bit ALU

Explanation of Terms

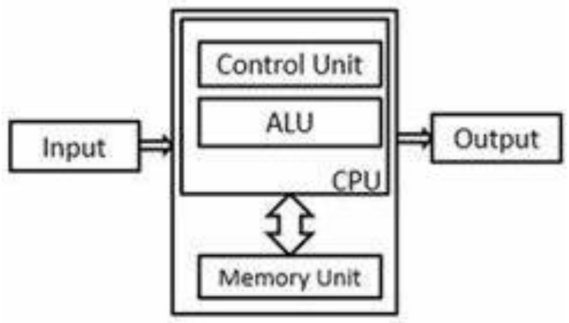
- *Bit size* refers to **the number of bits that can be processed simultaneously** by the basic arithmetic unit of a microcontroller. A number of bits taken as a group in this manner is called a ***word***.
  - For example, an 8-bit microcontroller can process an 8-bit word.
- *CPU* (Central Processing Unit) contains several registers (fast memory elements), an ALU, and a control unit.

Explanation of Terms

- *Harvard architecture* is a type of CPU architecture which uses separate program and data memory units along with separate buses for program and data.
  - It thus can execute programs and access data simultaneously.
  - Several microcontrollers including the PIC18F are designed using the Harvard architecture.
- Von Neumann architecture is another type of CPU architecture which uses a single shared memory and bus for both, offering simpler design but potentially leading to the von Neumann bottleneck.



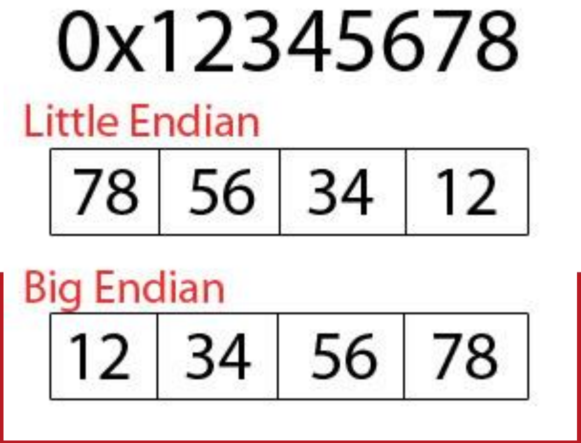
Harvard Model



Von Neumann Model

Explanation of Terms

- *Little endian*: the low memory address stores the low byte while the high memory address stores the high byte.
  - **The PIC18F microcontroller follows the little-endian format**.
- *Big endian* convention is used to store a 16-bit number such as 16-bit data in two bytes of memory locations as follows: the low memory address stores the high byte while the high memory address stores the low byte.



Explanation of Terms

- *Pipelining* is a technique that overlaps instruction fetch (instruction read) with execution. Pipelining is often used to fetch the microcontroller’s next instruction while executing the current instruction. PIC18F (8-bit microcontroller) uses a two-stage instruction pipeline.
- Random-access memory (RAM) is a read/write memory.
- Read-only memory (ROM) is a storage device whose contents can only be read.

Explanation of Terms

- *Reduced Instruction Set Computer* (RISC) contains a simple instruction set.
  - RISC microcontrollers need fewer transistors and enable a smaller die size of the integrated circuitry (IC).
    - RISC microcontrollers consume less power compared to CISC
    - Suitable for portable devices.
    - Microchip’s P IC18F is a RISC-based microcontroller.
- *Complex Instruction Set Computer* (CISC) contains a large instruction set.
  - NXP/ Freescale/Motorola HC11 is a CISC-based microcontroller.

Unsigned and Signed Binary Numbers

- An *unsigned binary number* is always positive. An 8-bit unsigned binary integer represents all numbers from 00_{16} through FF_{16} (0_{10} through 255_{10}).
- A *signed binary number* includes both positive and negative numbers. The decimal number +15 is represented in 8-bit two's-complement form as 00001111 (binary) or 0F (hexadecimal). The decimal number -15 can be represented in 8-bit two's-complement form as 11110001 (binary) or F1 (hexadecimal).

正數和0的二補數就是該數字本身。負數的二補數則是將其對應正數按位元取反再加1。

Unsigned and Signed Binary Numbers

- A *signed binary number*
  - The most significant bit (MSB) of a signed number represents the sign of the number.
  - A “0” at the MSB represents a positive number; a “1” at the MSB represents a negative number. Note that the 8-bit binary number 11111111 is 255_{10} when represented as an unsigned number. On the other hand, 11111111_2 is -1_{10} when represented as a signed number.

Unsigned and Signed Binary Numbers

- One can convert an unsigned binary number from lower to higher length using zero extension.
  - For example, an 8-bit unsigned number FF (hex) can be converted to a 16-bit unsigned number 00FF (hex) by extending 0's to the upper byte of 00FF (hex).
  - Both FF (hex) and 00FF (hex) have the same decimal value of 255.
  - Zero extension is useful for performing arithmetic operations between two unsigned binary numbers of different lengths.

Unsigned and Signed Binary Numbers

- A signed binary number, on the other hand, can be converted from lower to higher length using *sign extension*.
  - For example, an 8-bit signed number FF (hex) can be converted to a 16-bit signed number FFFF (hex) by extending the sign bit ('1' in this case) to the upper byte of FFFF (hex).
  - Both FF (hex) and FFFF (hex) have the same decimal value of -1.

Unsigned and Signed Binary Numbers

- Sign extension is useful when one wants to perform an arithmetic operation on two signed numbers of different lengths. For example, the 16-bit signed number 0020 (hex) can be added with the 8-bit signed number E1 (hex) by sign-extending E1 as follows:

$$\begin{array}{lr} 0020_{16} = & 0\,0\,0\,0\ \ 0\,0\,0\,0\ \ 0\,0\,1\,0\ \ 0\,0\,0\,0\ (32_{10}) \\ \text{Sign extension} \quad E1_{16} = & \boxed{1\,1\,1\,1\ \ 1\,1\,1\,1}\ \ 1\,1\,1\,0\ \ 0\,0\,0\,1\ (-31_{10}) \\ \hline & 1\ \ \underbrace{0\,0\,0\,0}_{0}\ \ \underbrace{0\,0\,0\,0}_{0}\ \ \underbrace{0\,0\,0\,0}_{0}\ \ \underbrace{0\,0\,0\,1}_{1}\ (+1_{10}) \end{array}$$

Ignore carry $\nearrow$ 1

Unsigned and Signed Binary Numbers

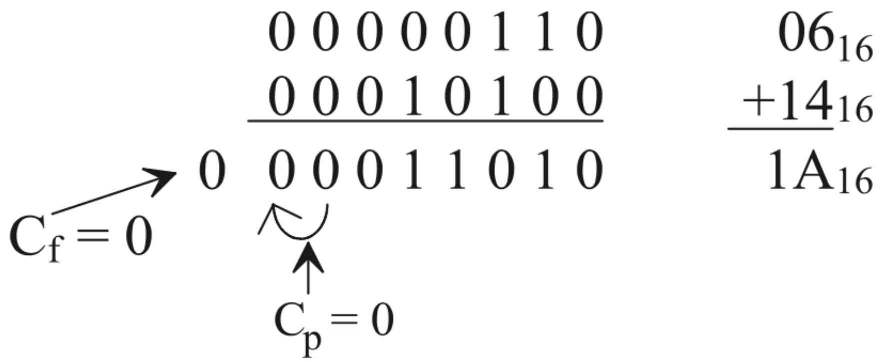
- An error (indicated by overflow in a microcontroller) may occur while performing two’s complement arithmetic.
- The microcontroller automatically sets an *overflow bit* to 1 if the result of an arithmetic operation is too big for the microcontroller’s maximum word size; otherwise, it is cleared to 0.

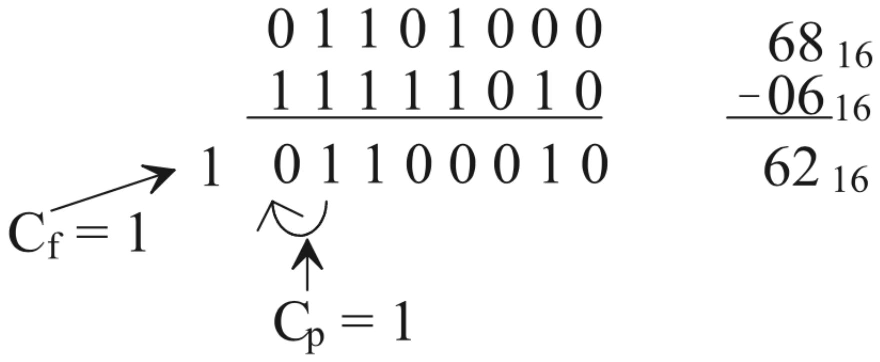
Unsigned and Signed Binary Numbers

- Consider two 8-bit signed numbers. Let C_f be the final carry (carry out of the most significant bit or sign bit) and C_p be the previous carry (carry out of bit 6 or seventh bit).
- We will show by means of numerical examples that
  - **As long as C_f and C_p are the same, the result is always correct.**
  - **If C_f and C_p are different, the result is incorrect and sets the overflow bit to 1.**

Unsigned and Signed Binary Numbers

Case 1: C_f and C_p are the same.



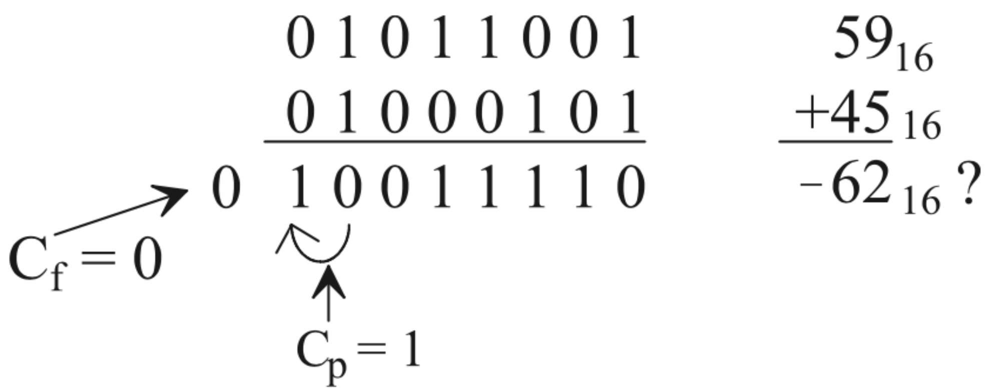


Therefore, when C_f and C_p are either both 0 or both 1, a correct answer is obtained.

國立成功大學 National Cheng Kung University 1931

Unsigned and Signed Binary Numbers

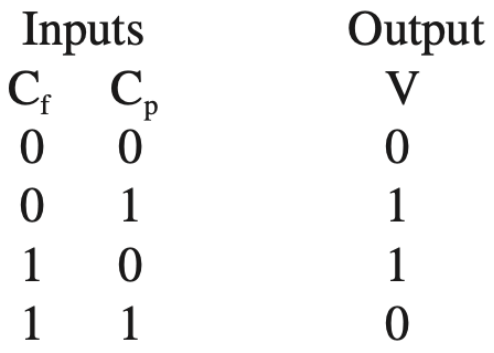
Case 2: C_f and C_p are different.



$C_f = 0$ and Cp = 1 give an incorrect answer because the result shows that the addition of two positive numbers is negative.

Unsigned and Signed Binary Numbers

- C_f = 1 and C_p = 0 provide an incorrect answer because the result indicates that the addition of two negative numbers is positive. Hence, the **overflow bit (V) will be set to zero if the carries C_f and C_p are the same**. On the other hand, the **overflow bit (V) will be set to 1 if carries C_f and C_p are different.** The relationship among C_f , C_p, and V can be summarized in a truth table as follows:



Inputs		Output
C_f	C_p	V
0	0	0
0	1	1
1	0	1
1	1	0

- From the truth table, overflow, $V = C_f \oplus C_p$

Unsigned and Signed Binary Numbers

- The PIC18F microcontroller includes unsigned multiplication instructions. However, the PIC18F does not provide any signed multiplication and division (both signed and unsigned) instructions.

ASCII and EBCDIC Codes

- A microcontroller must be capable of handling nonnumeric information. These codes are classified as alphanumeric or character codes. This totals to 87 characters. To represent 87 characters with some type of binary code would require at least 7 bits.
  - 26 lowercase letters
  - 26 uppercase letters
  - 10 numerical digits (0–9)
  - Approximately 25 special characters, which include +, /, #, %, and others.

ASCII and EBCDIC Codes

- The two most common alphanumerical codes are the American Standard Code for Information Interchange (ASCII) and the extended binary-coded-decimal interchange code (EBCDIC). ASCII is typically used with microprocessors; IBM uses EBCDIC code.
  - Alphanumeric codes (also known as character codes) are defined as binary codes used to represent alphanumeric data.

ASCII and EBCDIC Codes

- Note that decimal digits 0 through 9 are represented by 30_{16} through 39_{16} in ASCII.
- On the other hand, these decimal digits are represented by $F0_{16}$ through $F9_{16}$ in EBCDIC
  - EBCDIC code is obsolete
- A microcontroller program is usually written for code conversion when input/ output devices of different codes are connected to the microcontroller.

Unpacked and Packed Binary-Coded-Decimal Numbers

- The 10 decimal digits 0 through 9 can be represented by their corresponding 4-bit binary numbers. The digits coded in this fashion are called *binary-coded-decimal digits* in 8421 code, or BCD digits.
- Two unpacked BCD bytes are usually packed into a byte to form packed BCD. For example, two unpacked BCD bytes 02_{16} and 05_{16} can be combined as a packed BCD byte 25_{16}.

Unpacked and Packed Binary-Coded-Decimal Numbers

- Let us consider entering data decimal 24 via an ASCII keyboard. Two keys (2 and 4) will be pushed. This will generate 32 and 34 (32 and 34 are ASCII codes in hexadecimal for 2 and 4, respectively) inside the microcontroller. A program can be written to convert these ASCII codes into unpacked BCD 02_{16} and 04_{16}. This data can be converted to packed BCD 24 or to binary.

Unpacked and Packed Binary-Coded-Decimal Numbers

- Unpacked BCD 02_{16} and 04_{16} can be converted into packed BCD 24 (00100100_2) by logically shifting 02_{16} to obtain 20_{16} , then logically ORing with 04_{16}. On the other hand, to convert unpacked BCD 02_{16} and 04_{16} into binary, one needs to multiply 02_{16} by 10 and then add 04_{16} to obtain 00011000_2 (the binary equivalent of 24).

Unpacked and Packed Binary-Coded-Decimal Numbers

- Note that BCD correction (adding 6) is necessary for the following:
  - i) If the binary sum is greater than or equal to decimal 10 (This will generate a carry of one).
  - ii) If the binary sum is 1010 through 1111.
- For example, consider adding packed BCD numbers 97 and 39:

$$
\begin{array}{rrrl}
 & 111 \leftarrow \text{Intermediate Carries} & & \\
97 & 1001 & 0111 & \text{BCD for 97} \\
\underline{+39} & 0011 & 1001 & \text{BCD for 39} \\
\hline
136 & 1101 & 0000 & \text{invalid sum} \\
 & +0110 & +0110 & \text{add 6 for correction} \\
\hline
\underbrace{0001}_{1} & \underbrace{0011}_{3} & \underbrace{0110}_{6} & \leftarrow \text{correct answer 136}
\end{array}
$$

BCD碼的主要優點是在機器格式與人可讀的格式之間轉換容易，以及十進位數值的高精度表示。BCD碼的主要缺點是增加了實現算術運算的電路的複雜度，以及儲存效率低。

Unpacked and Packed Binary-Coded-Decimal Numbers

- Typical 32-bit microprocessors such as the NXP/Freescale/Motorola 68020 include PACK and UNPK instructions for converting an unpacked BCD number to its packed equivalent, and vice versa.
- The PIC18F microcontroller contains an instruction called DAW which provides the correct BCD result after binary addition of two packed BCD numbers.