Microcomputer Technology
PC Hardware

- **IBM PC introduced in 1981**
- **PC made legitimate by association with mainframe vendor IBM**
- **PC began as 16-bit Intel 8088 chip**
- **PC evolved into 32-bit machines using Pentium chip and clone chips**

Microcomputer Technology
PC Chip History

YEAR	CHIP	MAX. CLOCK SPEED	DATA TYPES	DATA PATH	MAX. MEMORY SIZE	VIRTUAL MEMORY
1978	8086	4 mhz	8, 16 bits	16 bits	1 MB	NO
1979	8088	8	8, 16	8	1	NO
1982	80286	16	8, 16	16	16	YES
1986	80386	40	8, 16, 32	32	4 GB	YES
1989	80486	90	8, 16, 32	32	4 GB	YES
1993	Pentium	200	8, 16, 32	32, 64	4 GB	YES
1995	Pentium Pro	300	8, 16, 32	32, 64	4 GB	YES
1997	Pentium II	500	8, 16, 32	32, 64	4 GB	YES
1999	Pentium III	700+	8, 16, 32	32, 64	4 GB	YES

Microcomputer Technology
PC Operating Systems

- CP/M: 8-bit O.S., forerunner of MSDOS
- MSDOS: cobbled from CP/M and UNIX
- UNIX: adapted from mainframe UNIX
- LINUX: unix-like, developed for PC
- MS Windows: GUI (versions 3.0, 95, 98, 2000)
- MS Windows NT: client/server O.S.

Microcomputer Technology Architecture Fundamentals

- Microprocessor: programmable logic device
- Microcomputer: interconnected microprocessor, memory and Input/Output
- Microcomputer organization:
    - control unit: coordinates instruction execution
    - memory unit: stores data, instructions
    - arithmetic unit: executes arithmetic and logic
    - I/O unit: communicates with outside world
    - Buses: connects the above elements
    - Stack: temporary storage of data for operations in progress

Microprocessor: A Closer Look

- Program counter
- Instruction register
- Instruction decoder
- Data address register
- Data, address buses
- Stack
- Scratch pad (accumulator)
- Arithmetic logic unit
- Cycle timing

Instruction Format

- **Instruction cycle**
  - **Fetch and decode each instruction**

- **Execution cycle**
  - **Fetch needed data and execute each instruction**

Operation code	Operand 1	Operand 2
Add	**ax**	**bx**

add register **ax** to register **bx**, place sum in **ax**
machine code = 01D8h

Inside the PC
Major Chip Functions

- **Microprocessor: multiple use for some pins**
- **Timing diagram: valid periods for information**
- **Coprocessors: math, video, communications**
- **Interrupt controller: IRQs (interrupt requests)**
- **DMA Controller: direct memory access**
- **Clock generator: timing for all chip functions**
- **Programmable Peripheral Interface (PPI): manages I/O functions**
- **Video controller: generates video display**
- **RAM: random access memory**
- **ROM: read only memory**

Inside the PC
Memory RAM

- **Dynamic RAM: main memory, must be refreshed**
  - **FPM RAM: fast page memory, early chips, 30 mhz bus speed**
  - **EDO RAM: enhanced data-out, 66 mhz**
  - **BEDO RAM: burst enhanced data-out, multiple data elements sent, 66 mhz**
  - **SD RAM: synchronous dynamic, 100+ mhz**

- **Static RAM: cache memory**
  - **SRAM: static, faster, more expensive than DRAM**
  - **PBSRAM: pipeline burst static, multiple requests in a single burst**

Inside the PC
Video and CMOS RAM

- **Video RAM: supports the video display**
  - **VRAM: video, dual ports to refresh and write to the display simultaneously**
  - **WRAM: windows, optimized for video graphics**
  - **SGRAM: synchronous graphics, handles 2 memory pages at once**

- **CMOS RAM: stores setup information, refreshed by a small battery**

Inside the PC
Memory Connections and Port Addresses

- **Memory connections to the motherboard**
  - **SIMM: single inline memory module, packaging of DRAM, 32-bit data path**
  - **DIMM: dual inline memory module, packaging of DRAM, 64-bit data path**

- **I/O ports: addresses reserved to identify physical devices such as hard drives, printers, etc.**

Inside the PC
Bus Architecture

- **PC AT: introduced with 80286, 8-bit path 8mhz speed**
- **ISA: industry standard, 16-bit, 8 mhz, still used**
- **EISA: extended ISA, 32-bit**
- **MCA: IBM microchannel, proprietary, not used today**
- **VESA: video electronics standards association, high speed graphics with 32 or 64-bit path**
- **PCI: peripheral component interconnect, 32 or 64-bit path, up to 100 mhz, 500 MBs per second**
- **CPU local bus on the motherboard: 64-bit path, 100+ mhz**

Inside the PC
Secondary Storage Devices

- **Device controllers for floppy, CD-ROM and hard drives**
  - **IDE: intelligent device electronics, control electronics on motherboard or an adapter card**
  - **EIDE: enhanced IDE, larger drives, faster transfer speed (33 MB per sec)**
  - **SCSI: small computer system interface, controls multiple devices -- disks, scanners, printers**

PC System Architecture
Registers

- **General purpose registers**
  - **AX: accumulator, used in many arithmetic functions**
  - **BX: base, arithmetic and addressing functions**
  - **CX: counter, for looping instructions**
  - **DX: data, important for multiply and divide**

- **Segment registers: base locations for program instructions**
  - **CS: code segment, executable instructions**
  - **DS: data segment, variables**
  - **SS: stack segment, stack contents**
  - **ES: extra segment, additional base for variables**

PC System Architecture
Registers

- **Index registers: contain offsets from base locations**
  - **SI: source index, source for string movement instructions, offset from DS**
  - **DI: destination index, destination for string movement instructions, offset from ES**
  - **BP: base pointer, variable locator, offset from SS**
- **Special registers: contain offsets**
  - **IP: instruction pointer, offset of the next instruction**
  - **SP: stack pointer, offset from beginning of the stack (SS)**
- **Flags register: shows CPU and results of arithmetic operations using individual bits**

PC System Architecture
Stack

- **Stack: used for temporary storage for addresses and data, reserved locations in RAM**

- **Starting location: high memory location**

- **Ending location: lower memory location**

- **Add to the stack: “push” a value on**

- **Remove from the stack: “pop” a value off**

PC System Architecture
Segmented Addressing

- **Linear addressing:**

  locations numbered 1, 2, … n,

  example: address = 21340h

- **Segmented addressing:**

  locations numbered as segment + offset

  example: instruction address = 12340h + 4321h = 16661

  where CS = 12340h and IP = 4321h

  maximum offset value/segment size = FFFFh = 65,535d

  maximum segment value = FFFFFh = 1,048,575d

PC System Architecture
Segmented Address Notation and Calculation

- **Instruction addresses are denoted: CS:IP**
  for example, 1234:4321

- **To calculate an absolute address location, multiply CS by 16 and add IP as**

  12340 + 4321 = 16661

PC System Architecture
Segmented Address Notation and Calculation

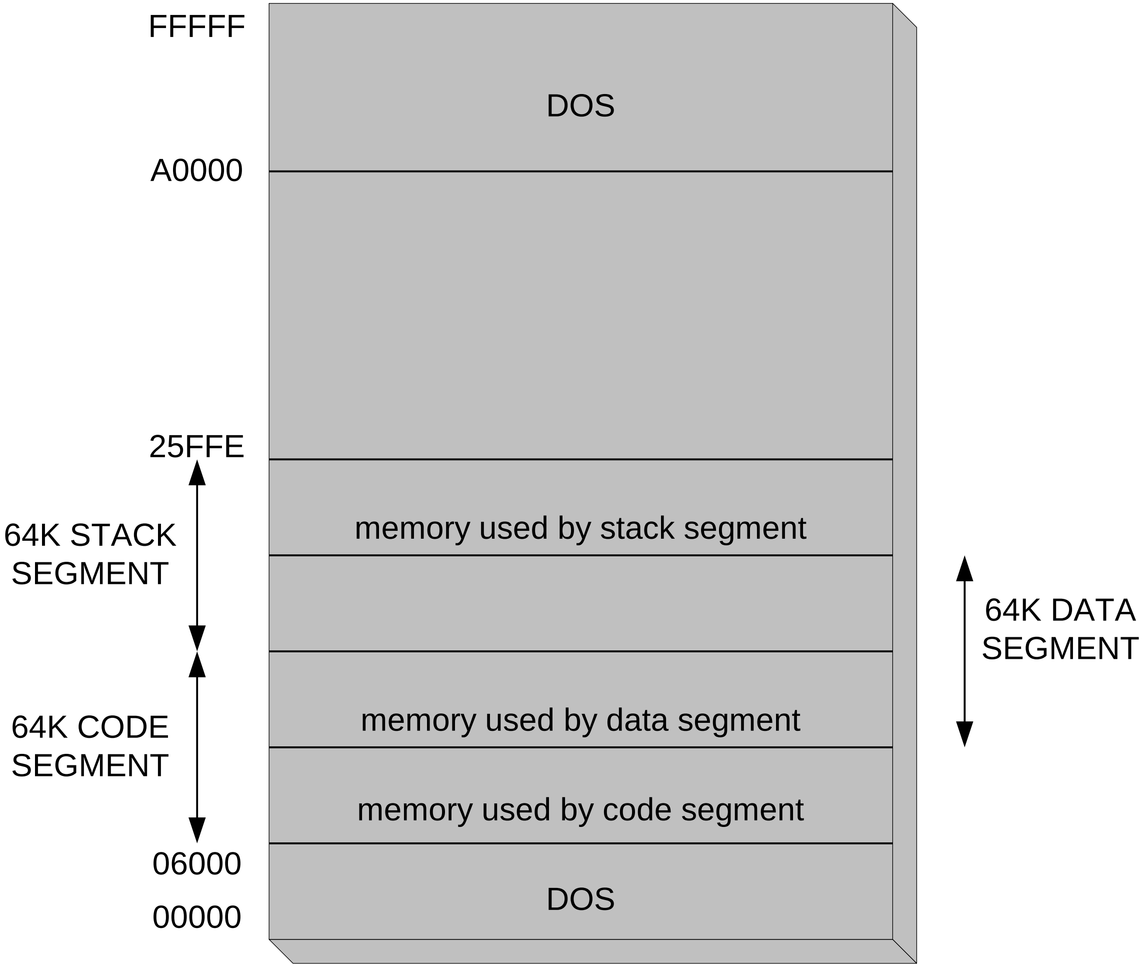
- One absolute location has many equivalent segmented address designations

- For example: 0000:0020 and 0001:0010 specify the same absolute address

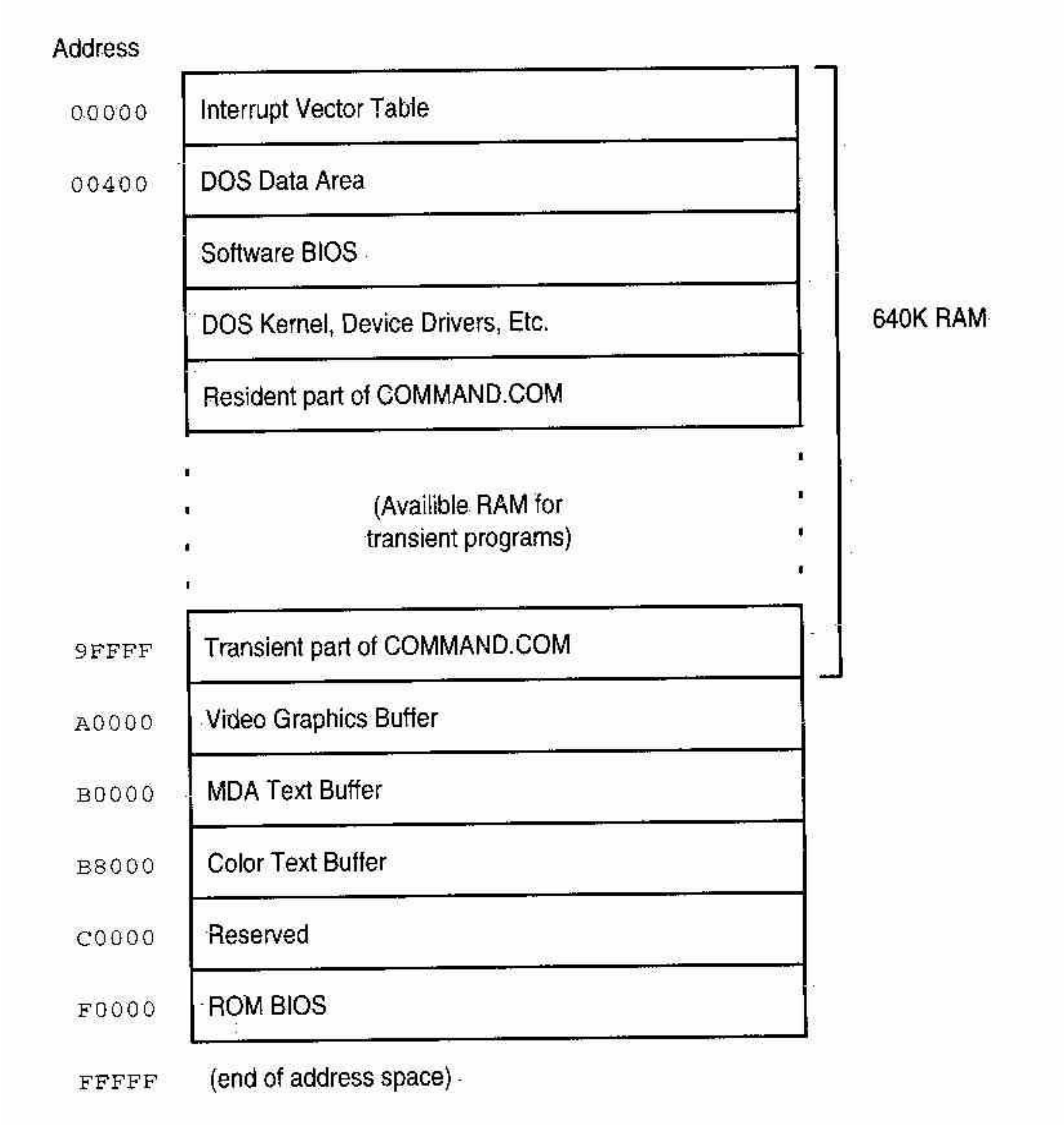
  since: 00000 + 0020 = 00020 and

  00010 + 0010 = 00020

PC System Architecture
Segmented Memory Map


FFFFF
DOS
A0000
25FFE
memory used by stack segment
64K STACK SEGMENT
64K DATA SEGMENT
memory used by data segment
64K CODE SEGMENT
memory used by code segment
06000
DOS
00000

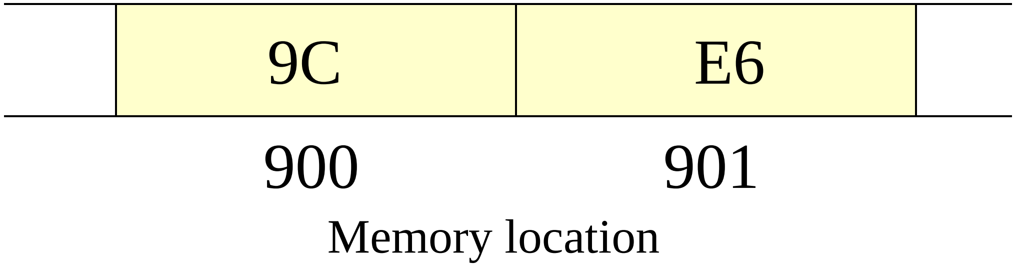
PC System Architecture
Memory Map



Address	
00000	Interrupt Vector Table
00400	DOS Data Area
	Software BIOS
	DOS Kernel, Device Drivers, Etc.
	Resident part of COMMAND.COM
	(Availible RAM for transient programs)
9FFFF	Transient part of COMMAND.COM
A0000	Video Graphics Buffer
B0000	MDA Text Buffer
B8000	Color Text Buffer
C0000	Reserved
F0000	ROM BIOS
FFFFF	(end of address space)

PC System Architecture
“Backwords” Storage or “Little Endian” Ordering

Consider a 2-byte word in memory:



9C E6

900 901

Memory location

Q. Is the 2-byte word 9C E6 or E6 9C?

A. It depends

In a “little endian” machine like the PC, the less significant byte is stored in the lower memory location.

Hence the word is: E6 9C and it appears to be stored “backwords”

Evolution of PC System Architecture

<u>8088/8086, 80286 PCs</u>

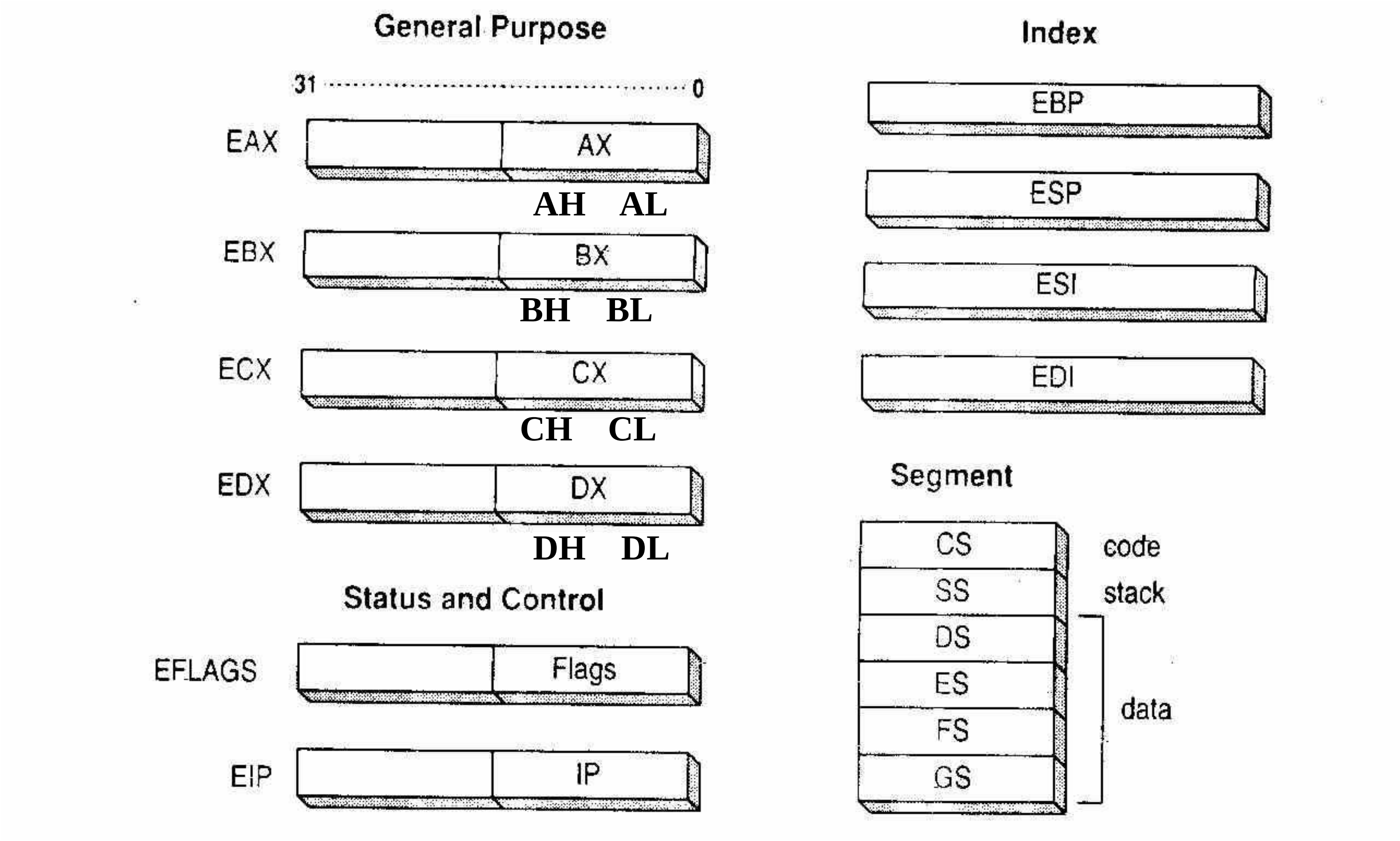
- **16-bit word, registers**
- **40,000 - 130,000 transistors**
- **clock speed: 5 - 16 mhz**
- **max. memory: 1- 16 MB**
- **segmented memory, max. segment = 64K bytes**
- **some instruction pipelining**
  (4 - 6 byte instruction queue)
- **rudimentary multitasking**

Evolution of PC System Architecture

<u>**80386, 80486 PCs**</u>

- **32-bit word, registers**
- **275,000 - 1,000,000 transistors**
- **clock speed: 40 -90 mhz**
- **max. memory: 4 GB**
- **segmented memory, max. segment: 4 GB (linear addressing)**
- **improved instruction pipelining (16 - 32 byte instruction queue)**
- **multitasking with 64 TB per task**

32-Bit Register Set



Evolution of PC System Architecture

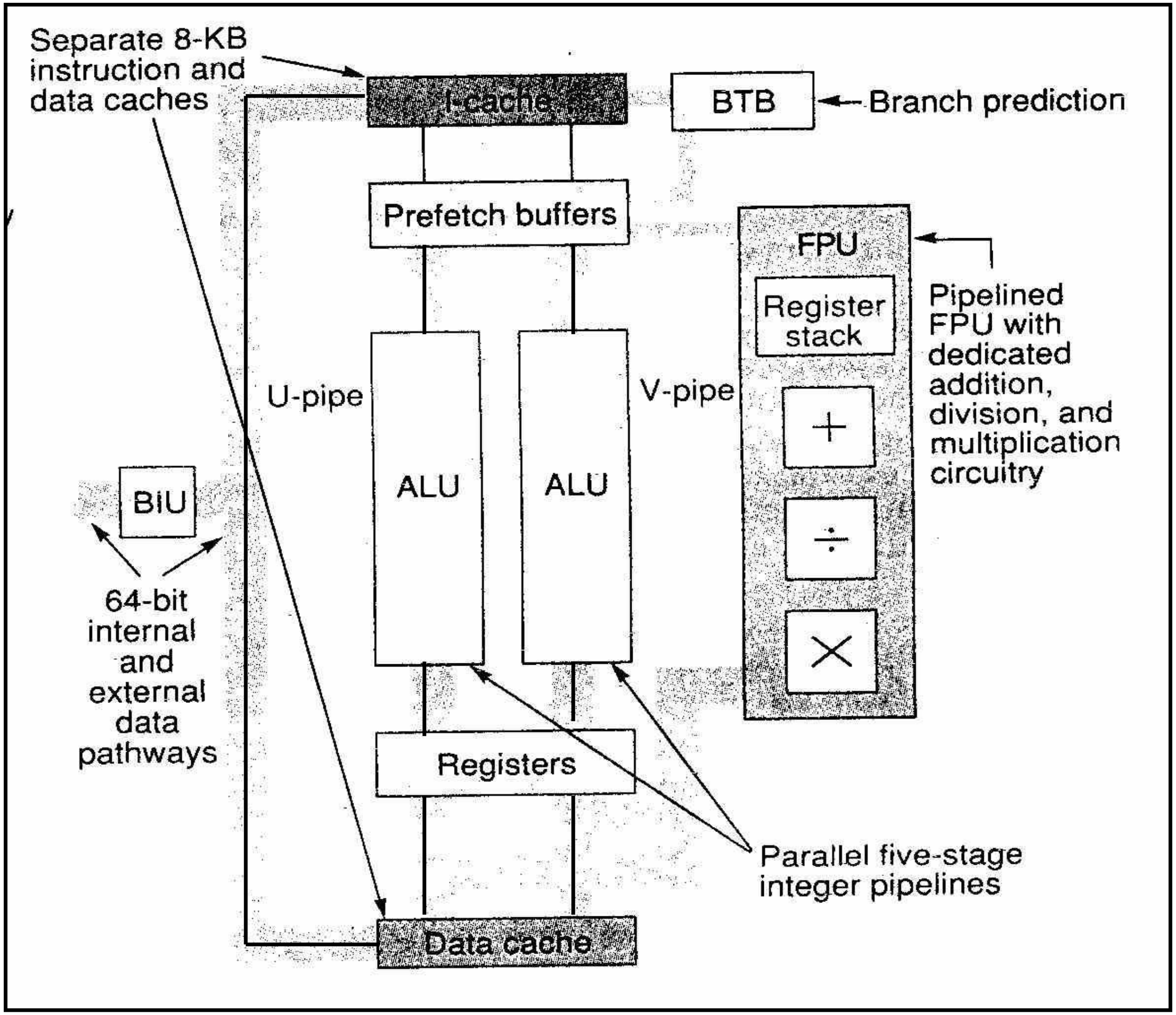
Pentium family PCs

- **32-bit word, registers**
- **3,000,000 - 15,000,000 transistors**
- **clock speed: 200 -700+ mhz**
- **max. memory: 4 GB**
- **segmented memory, max. segment: 4 GB**

  (linear addressing)
- **multiple pipelining with branch prediction**

  (16 - 32 byte instruction queue)
- **super scaler execution providing multiple instructions per clock**
- **64-bit bus**

Pentium Family Architecture


Separate 8-KB instruction and data caches
I-cache
BTB
Branch prediction
Prefetch buffers
FPU
Register stack
Pipelined FPU with dedicated addition, division, and multiplication circuitry
U-pipe
V-pipe
ALU
ALU
+
÷
×
BIU
64-bit internal and external data pathways
Registers
Parallel five-stage integer pipelines
Data cache

Positional Number Systems

Consider the base 10 number: 431

It can be expressed as:

$$4 \times 10^2 + 3 \times 10^1 + 1 \times 10^0 =$$
$$4 \times 100 + 3 \times 10 + 1 =$$
$$400 + 30 + 1 =$$
$$431$$

The numbers: 4, 3 and 1 are coefficients and the powers of 10 (100, 10 and 1) are positional weights or place values.

All positional number systems share this property regardless of the base or radix used. Other common bases include 2, 8 and 16.

Positional Number Systems
Integers

In general the value, V, of a number in the positional number system is represented as follows.

$$V = \sum_{i=0}^{m} c_i \, b^i$$

where: the c_i are the coefficients, b is the base or radix, and m is one less than the number of digits in the number.

For the previous example we have:

$$431 = \sum_{i=0}^{2} c_i b^i = 1 \times 10^0 + 3 \times 10^1 + 4 \times 10^2 = 1 + 30 + 400$$

Positional Number Systems
Floating Point Numbers

To define a decimal or, floating point number, V is represented as follows:

$$V = \sum_{i=-n}^{m} c_i \, b^i$$

where: the c_i are the coefficients, b is the base or radix,

m is one less than the number of digits in the number,

and n is the number of decimal places.

For example, the floating point number 431.25 is:

$$431.25 = \sum_{i=-2}^{2} c_i b^i = 5 \times 10^{-2} + 2 \times 10^{-1} + 1 \times 10^{0} + 3 \times 10^{1} + 4 \times 10^{2}$$

$$= 5 \times .01 + 2 \times .10 + 1 + 3 \times 10 + 4 \times 100$$

$$= .05 + .20 + 1 + 30 + 400$$

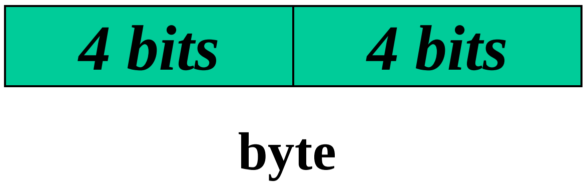
Positional Number Systems

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

Positional Number Systems

Why use such an “odd” base as hexadecimal?

Computer words are divided into basic units called bytes, each byte consisting of 8 bits or binary digits.



4 bits ***4 bits***

byte

Each half of the byte can be represented as *one* hex character, so *two* hex characters can compactly represent the eight bits of a byte in a memory display.

Notation for Number Systems

Is 10 a number of base 2, base 10 or base 16?

- **10b denotes base 2 or binary**

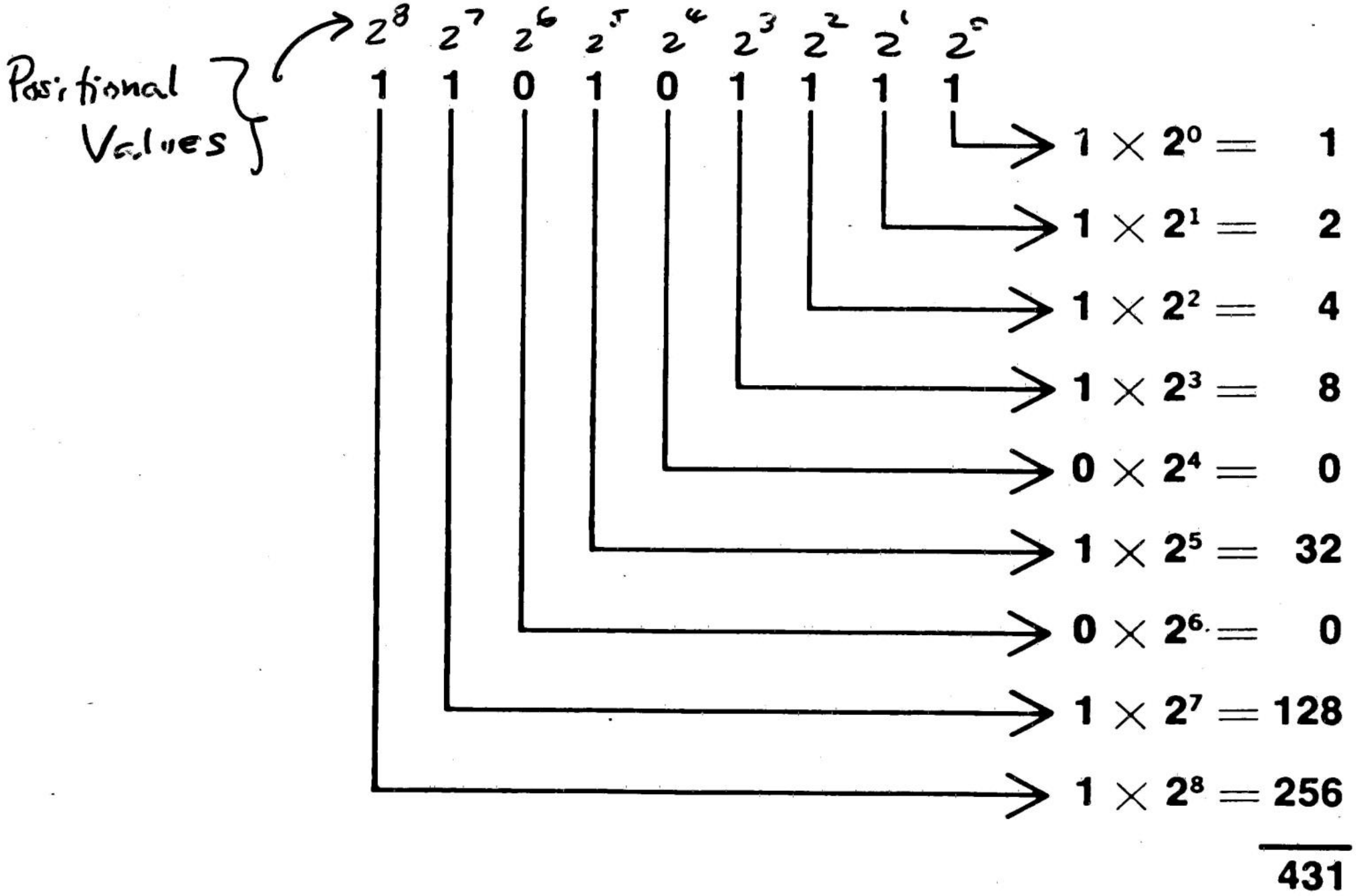
- **10d denotes base 10 or decimal**

- **10h denotes base 16 or hexadecimal**

Note that 10b = 2d, 10d = 10d and 10h = 16d

Thus, 10 can represent three different values!

Positional Number Systems
Binary to Decimal Conversion



$$1 \times 2^8 + 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 431$$

Data Formats

CHARACTER - 7, 8 OR 16 BIT REPRESENTATION

- **ASCII (American standard code for information interchange):**

  7 bit standard

- **ASCII augmented: 8 bit defacto standard**

- **EBCDIC (extended binary coded decimal interchange code):**

  8 bit defacto standard

- **UNICODE: 16 bit to accommodate foreign languages**

Data Formats

INTEGER (unsigned) binary representation

Given n bits, (usually a multiple of 8), we can represent numbers in the range 0 to 2^n-1.

If n=8 (byte), range = 0 to 2^8-1 = 0 to 255, or 0 to FF.

If n=16 (word), range = 0 to 2^{16}-1 = 0 to 65,535, or 0 to FFFF

If n=32 (double word), range = 0 to 2^{32} - 1 =

0 to 4,294,967,295, or 0 to FFFFFFFF

Data Formats

INTEGER (signed) binary representation

Requires 1 bit for the sign. Given n bits, we can represent numbers in the

range $1-2^{n-1}$ to $+2^{n-1}-1$.

If n=8 (byte), range = $1-2^7$ to $+2^7-1$ = -127 to +127

But, signed numbers are represented more functionally in two's complement

form, where the range is from -2^{n-1} to $+2^{n-1}-1$.

If n=8 (byte), range = -2^7 to $+2^7-1$ = -128 to +127 or 80, 81,…,FF, 0, 1, 2,…,7F

If n=16 (word), range = -32,768 to +32,767

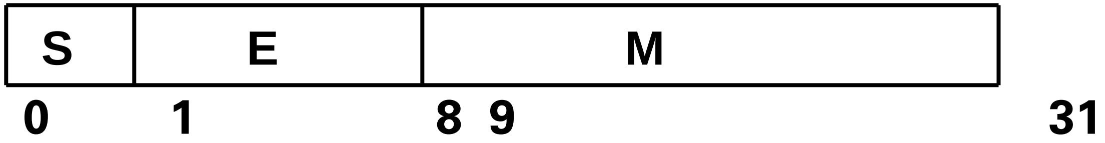
or 8000, 8001,…,FFFF, 0, 1, 2,…, 7FFF

If n=32 (double word), range is approximately ±2.147 x 10^9

Data Formats

FLOATING POINT - IEEE 754 STANDARD

For a 32 bit word, the scheme is as follows:



S E M

0 1 8 9 31

S is 1 bit for sign, 0 or 1 , E is 8 bits for exponent, M is 23 bits for mantissa

Convention for M uses an implicit 1 before M. For example: 00100110 is stored as M = 0011. Trailing zeros are also ignored.

Value of the floating point number is calculated as:

N = $(-1)^{S}(2^{E-127})(1.M)$, where N is floating point binary.

Note sign determination: $(-1)^{0} = 1$, so S = 0 indicates +; $(-1)^{1} = -1$, so S = 1 indicates -

Data Formats

FLOATING POINT - IEEE 754 STANDARD (CONT'D.)

Note the exponent is excess 127, and minimum value of E is 0, and the maximum value of E is 2^8-1 or 255.

Therefore, the exponent ranges from 2^{0-127} to $2^{255-127}$ or 2^{-127} to 2^{128}, where the power of 2 in the exponent determines the number of places to move the binary point to the left (-) or right (+).

EXAMPLE 1:

```
0    1000 0000       011 0000 0000 0000 0000 0000
sign  exponent          mantissa
```

$N=(-1)^0(2^{128-127})(1.011)=(1)(2^1)(1.011)=10.11$

$(10.11)_2=(2.75)_{10}$

Two's Complement Arithmetic

Introduction

The two's complement form of numbers is used to facilitate addition

and subtraction of signed numbers. Two's complement form allows this

to be accomplished using only addition. Two's complement form is

defined as follows.

If X_{10} is a base 10 number, the n-bit two's complement form (TCF) is:

TCF = $(X_{10})_2$ with a 0 bit appended to the left if $X_{10} > 0$

which may be written as: TCF = $0,(X_{10})_2$

TCF = $[2^{n-1} - |X_{10}|]_2$ with a 1 bit appended if $X_{10} < 0$

which may be written as: TCF = $1,[2^{n-1} - |X_{10}|]_2$

Two's Complement Arithmetic

EXAMPLES

Let $n=4$ and $X_{10}=7$. TCF = 0,111

Let $n=4$ and $X_{10}=-7$. TCF = 1, $[2^3 - |-7|]_2 = 1,001$

The range of X_{10} is then: $-2^{n-1} < X_{10} < 2^{n-1} - 1$.

If $n=4$, then : $-8 < X_{10} < +7$

Two's Complement Arithmetic

The complete set of TCF's for this range is as follows:

(X_{10})	(TCF)	
SIGNED DECIMAL	TWO'S COMPLEMENT	SIGNED MAGNITUDE
EQUIVALENT	FORM*	REPRESENTATION*
+7	0,111	0,111
+6	0,110	0,110
+5	0,101	0,101
+4	0,100	0,100
+3	0,011	0,011
+2	0,010	0,010
+1	0,001	0,001
0	0,000	0,000 AND 1,000
-1	1,111	1,001
-2	1,110	1,010
-3	1,101	1,011
-4	1,100	1,100
-5	1,011	1,101
-6	1,010	1,110
-7	1,001	1,111
-8	1,000	------

*** SIGN BIT: 0, = + ; 1, = -**

Two's Complement Arithmetic

Evaluating Two's Complement Numbers

Given a TCF: d_{n-1} d_{n-2} ... d_0, where d_i are 1 or 0 bits, the decimal equivalent of the TCF is:

$$-2^{n-1}d_{n-1} + 2^{n-2}d_{n-2} + ... + 2^0d_0$$

If n=4 the expression is:

$$-2^3d_3 + 2^2d_2 + 2^1d_1 + 2^0d_0$$

EXAMPLES:

TCF = 1,001 Decimal equivalent = -8 + 0 + 0 + 1 = -7

TCF = 0,101 Decimal equivalent = 0 + 4 + 0 + 1 = +5

TCF = 1,111 Decimal equivalent = -8 + 4 + 2 + 1 = -1

Two's Complement Arithmetic

Evaluating two's complement numbers

Short cuts with two's complement form:

Notice that we can also compute the TCF as follows:

X_{10}=-5 TCF = 1111 - 0101 + 1 = 1,011

This is the same as that computed using the formula:

TCF = $1,[2^3 - |-5|]_2 = 1,011$

So, the TCF may be computed by taking the one's complement (1111- 0101) and adding one to it. And the one's complement is simply a bit reversal of the number itself, as:

1111
- 0101
1010 which reverses 1's and 0's in 0101

Two's Complement Arithmetic

Evaluating two's complement numbers

This suggests an easy way to evaluate the value of a TCF for any negative number. Given 1,011 we know it is a negative number because of the 1 bit in the extreme left position.

To determine the value we may form the two's complement of 1,011. Complementing the complement returns the original number.

One's complement of 011:	100
Adding one:	+ 1
	101

or -5, where the "-" is inferred from the 1 bit in the left most position of 1,011.

Two's Complement Arithmetic

TCF for one-byte signed integers

For a one-byte integer, we need n=8. The range of values is:

$-2^7 < X_{10} < 2^7 - 1$ or $-128 < X_{10} < +127$

The complete table of values may be outlined as follows. We also add here

the hexadecimal form of the numbers, since this is the form displayed in

any memory dump.

Two's Complement Arithmetic

TCF for byte and word signed integers

			Byte	Word
X_{10}	X_{16}	TCF(binary)	TCF(hex)	TCF(hex)
127	7F	0,111 1111	7F	007F
126	7E	0,111 1110	7E	007E
.	.	.	.	.
.	.	.	.	.
0	00	0,000 0000	00	0000
-1	-01	1,111 1111	FF	FFFF
-2	-02	1,111 1110	FE	FFFE
.	.	.	.	.
.	.	.	.	.
-126	-7E	1,000 0010	82	FF82
-127	-7F	1,000 0001	81	FF81
-128	-80	1,000 0000	80	FF80

Two's Complement Arithmetic
Computing TCFs in Hex

- **Byte**

TCF_{16} = FF - ($|X_{16}|$ - 1)   for $X_{16} < 0$

Examples:

If X_{16} = -1, TCF_{16} = FF - (1 - 1) = FF

If X_{16} = -7F, TCF_{16} = FF - (7F - 1) = FF - 7E = 81

- **Word**

TCF_{16} = FFFF - ($|X_{16}|$ - 1)   for $X_{16} < 0$

Two's Complement Arithmetic

Some Examples Of Two's Complement Arithmetic

Assume n=4

$$\begin{array}{rr} 7 & 0,111 \\ -2 & 1,110 \\ \hline 5 & 10,101 \end{array}$$

TCF= 1's comp. + 1 = 1101 + 1 = 1110

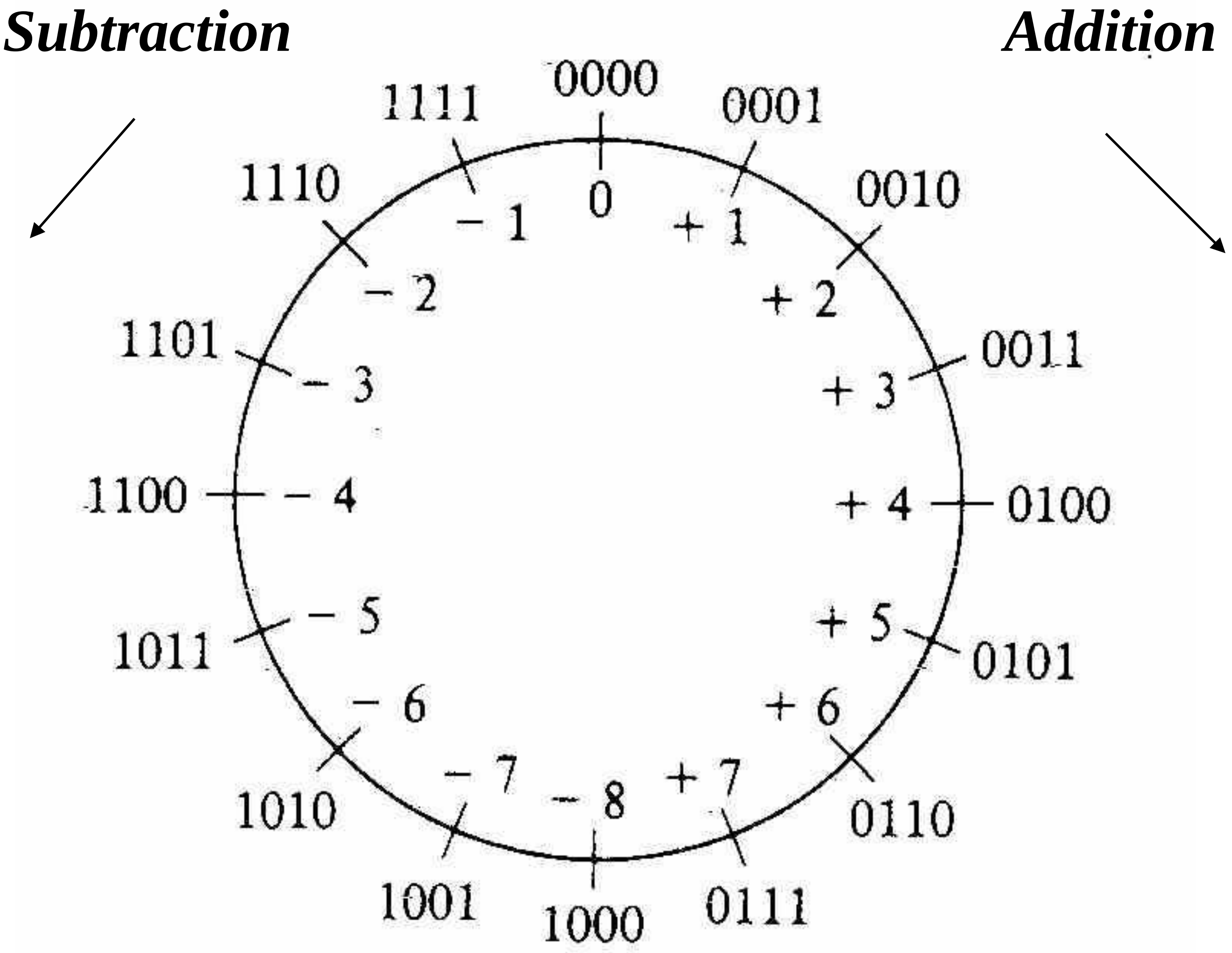
$$\begin{array}{rr} 2 & 0,010 \\ -7 & 1,001 \\ \hline -5 & 1,011 \end{array}$$

TCF = 1000 + 1 = 1001

Complementing the result, 011, we get: 100 + 1 = 101

or -5 since left most bit of 1,011 is 1.

Number Wheel



Subtraction

Addition

Instruction Format

Simplest Instruction Format

Operation code	Operand

The operation code indicates instruction to be executed, the while operand is

the entity to be operated upon. It may be a constant, a memory address, or

more than one memory address.

The F ving format.

Op code	Operand 1	Operand 2	Operand 3	. . .