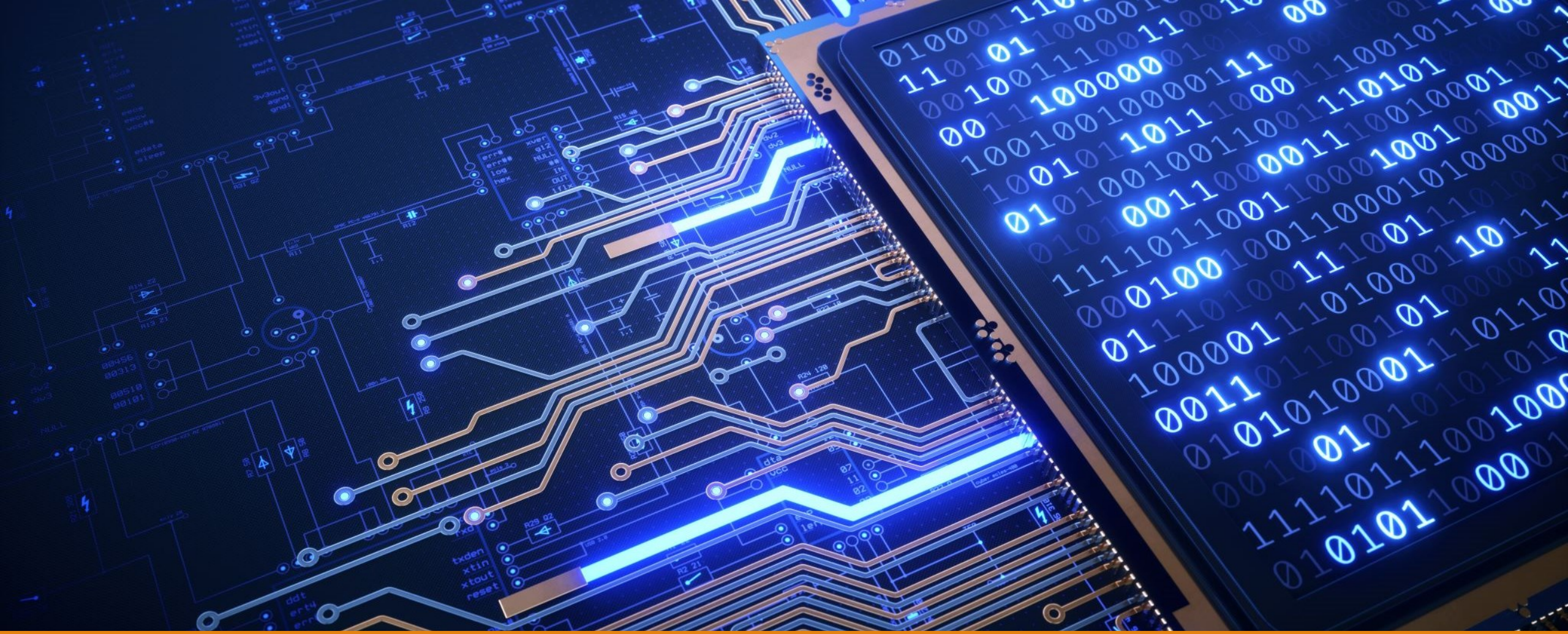


Microprocessor



1

Introduction

Outlines

1. Introduction
2. Important Definitions & Concepts
3. A Historical Background.
4. The Microprocessor-Based Computer System.
5. Number Systems.
6. Computer Data Formats.

Outlines

1. Introduction
2. Important Definitions & Concepts
3. A Historical Background.
4. The Microprocessor-Based Computer System.
5. Number Systems.
6. Computer Data Formats.

Outlines

1. Introduction
2. Important Definitions & Concepts
3. A Historical Background.
4. The Microprocessor-Based Computer System.
5. Number Systems.
6. Computer Data Formats.

Important Definitions & Concepts

- What is ***the Microprocessor*** (**µP**)?
  - It is the ***integrated circuit*** (IC) that represents the heart of a computer system.
  - Also called the ***CPU*** (**C**entral **P**rocessing **U**nit).
  - It is the controlling element in a computer system.
  - Controls the ***memory*** and ***I/O*** devoces through connections called ***buses***, using ***instructions*** stored in the memory.

Important Definitions & Concepts

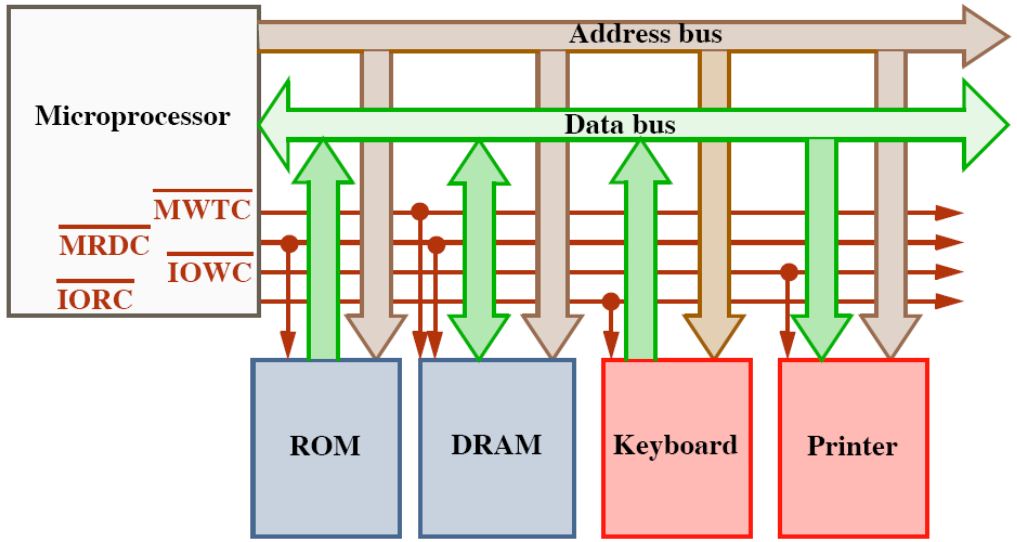
- What is ***the Microprocessor*** (**µP**)? (Cont.)
  - Performs **three main tasks**:
    - Data transfer between itself and the memory or I/O systems.
    - Simple arithmetic and logic operations.
    - Program flow via simple decisions.

Important Definitions & Concepts

- What is ***the Microprocessor*** (**µP**)? (Cont.)
  - **Three basic characteristics** differentiate microprocessors:
    - **Instruction set:** The set of instructions that the microprocessor can execute.
    - **Bandwidth:** The number of bits processed in a single instruction.
    - **Clock speed:** defines how many instructions per second the processor can execute.

Important Definitions & Concepts

- What are the ***buses***?
  - A common group of wires that interconnect components in a computer system.
  - Transfer address, data, & control information between microprocessor, memory and I/O.
  - Three buses exist for this transfer of information: address, data, and control.



Outlines

1. Introduction
2. Important Definitions & Concepts
3. A Historical Background.
4. The Microprocessor-Based Computer System.
5. Number Systems.
6. Computer Data Formats.

A Historical Background

- The Microprocessor Age:
  - World's ***first microprocessor*** is the ***Intel 4004***. (1971
    - A 4-bit microprocessor-programmable controller on a chip.
      - A **bit** is a binary digit with a value of one or zero.
      - **4-bit** wide memory location often called a **nibble.**
    - The 4004 instruction set contained 45 instructions.
    - Fabricated with P-channel MOSFET technology (state-of-the-art at that time).

A Historical Background

- The Microprocessor Age (Cont.):
  - World's ***first microprocessor*** is the ***Intel 4004***.
    - Executed instructions at 50 *KIPs* (**kilo-instructions per second**).
    - Slow compared to 100,000 instructions per second by 30-ton ENIAC computer in 1946.
    - The difference was that 4004 weighed less than an ounce.
    - Debuted in early game systems and small control systems.

A Historical Background

- The Microprocessor Age (Cont.):
  - ***The 8008 microprocessor (Intel 1971)***.
    - Extended **8-bit** version of 4004 microprocessor.
    - Expanded memory of 16K bytes.
      - A **byte** is generally an 8-bit-wide binary number and a **K** is 1024.
    - Memory size often specified in K bytes.
    - Contained additional instructions, 48 total.
    - Provided opportunity for application in more advanced systems.
      - Engineers developed demanding uses for 8008.
    - Somewhat small memory size, slow speed, and instruction set limited 8008 usefulness.

A Historical Background

- The Microprocessor Age (Cont.):
  - ***The 8080 microprocessor (Intel 1973).***
    - 8080 addressed four times more memory.
      - 64K bytes vs. l6K bytes for 8008.
    - Executed additional instructions; 10x faster.
      - Addition takes 20 µs on an 8008-based system, required only 2.0 µs on an 8080-based system.
    - TTL (transistor-transistor logic) compatible.
      - The 8008 was not directly compatible.
    - Interfacing made easier and less expensive.

A Historical Background

- The Microprocessor Age (Cont.):
  - ***The 8085 microprocessor (Intel 1977):***
    - Last 8-bit, general-purpose microprocessor .
    - Slightly more advanced than 8080; executed software at an even higher speed.
    - 769,230 instructions per second vs. 500,000 per second on the 8080.
    - Main advantages of 8085 were its internal clock generator and system controller, and higher clock frequency.
    - Higher level of component integration reduced the 8085's cost and increased its usefulness.

A Historical Background

- The Microprocessor Age (Cont.):
  - ***The Modern microprocessor, 8086 (Intel 1978), 8088 (Intel 1979)***:
    - **Higher speed and larger memory** size allowed 8086 & 8088 to replace smaller minicomputers in many applications.
    - Another feature was a **4- or 6-byte instruction cache or queue** that pre-fetched instructions before they were executed.
      - Queue sped operation of many sequences of instruction.
    - Basis for the much larger instruction caches found in modem microprocessors.

A Historical Background

- The Microprocessor Age (Cont.):
  - ***The Modern microprocessor, 8086 (Intel 1978), 8088 (Intel 1979):***
    - Increased memory size and additional instructions led to many efficient and sophisticated applications.
    - Improvements to the instruction set included **multiply** and **divide** instructions.
      - Missing on earlier microprocessors.
    - Number of instructions increased.
      - From 45 on the 4004, to 246 on the 8085.

A Historical Background

- The Microprocessor Age (Cont.):
  - ***The Pentium microprocessor (Intel 1993)***:
    - Intel decided not to use a number (in the naming) because it appeared to be impossible to copyright a number.
    - Originally named the **P5** or **80586**.
    - First versions were working at **60 MHz & 66 MHz**, and a speed of **110 MIPs**, while fastest version produced **233 MHZ Pentium** (a three and one-half the first version).
    - **Cache size** was increased to **16KB** vs. 8KB in 80486.
    - **Memory** system up to **4G bytes**, **data bus** width **64 bits**.

A Historical Background

- The Microprocessor Age (Cont.):
  - ***The Pentium Pro microprocessor (Intel 1996)***:
    - A recent release of Intel, **named P6**.
    - 3 integer units, floating-point unit, clock frequency of 150 and 166 MHZ.
    - **16K** level-one cache (**L1**), **256K** level-two cache (**L2**).
    - Executes up to three instructions at a time, execute in parallel.
    - Efficiently execute 32-bit code.
    - Can address **4G-byte or a 64G-byte memory** system.

A Historical Background

- The Microprocessor Age (Cont.):
  - ***The Pentium II microprocessor (Intel 1997)***:
    - Represents new direction for Intel.
    - On a small circuit board, instead of being an integrated circuit.
    - The microprocessor on the Pentium II module actually Pentium Pro with MMX extensions.

A Historical Background

- The Microprocessor Age (Cont.):
  - ***The Pentium III microprocessor (Intel 1999)***:
    - Faster core than Pentium II.
    - **32-bit microprocessor,**
    - **64GB** main **memory**.
    - 800MHZ and above.
    - **On-chip 256KB L2 cache (at-speed).**
    - Memory transfers 100MHz to **133MHz.**
    - **Dual Independent Bus (simultaneous L2 and system memory access).**

A Historical Background

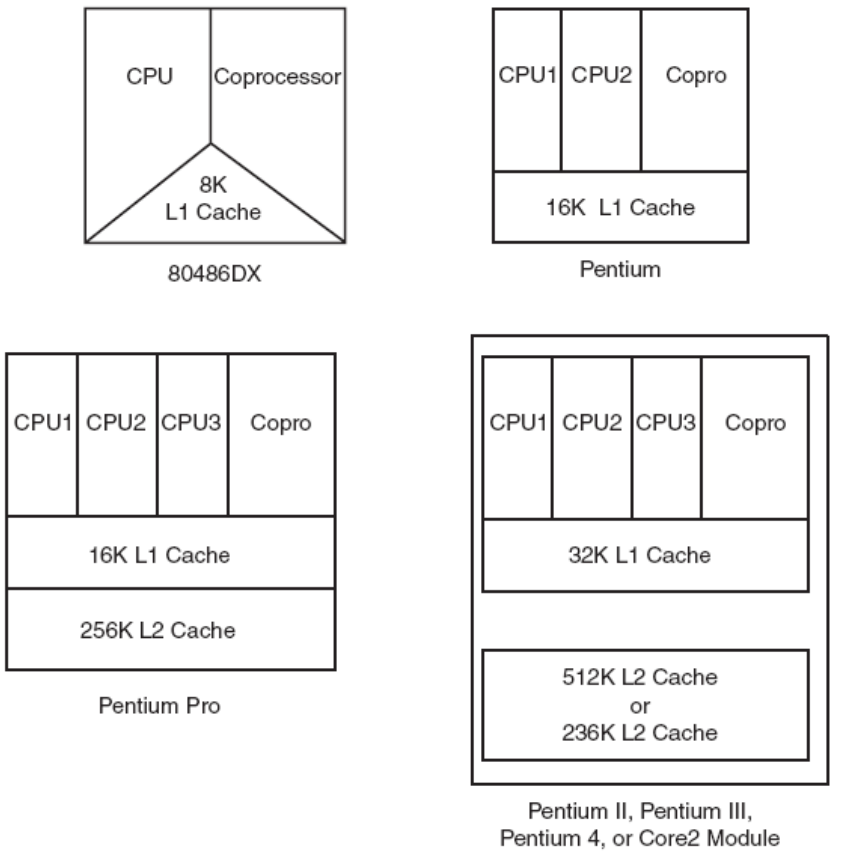
- The Microprocessor Age (Cont.):
  - ***The Pentium 4 microprocessor (Intel 2000)***:
    - Uses Intel P6 architecture.
    - Up to 3.2 GHz and faster.
    - Supporting chip sets that use RAMBUS or DDR memory instead of SDRAM technology.

A Historical Background

- The Microprocessor Age (Cont.):
  - ***Biggest advancement is the inclusion of multiple cores i3, i5, i7:***
    - Each core executes a separate task in a program.
    - Increases the speed of execution if program is written to take advantage of multiple cores which is called "***multithreaded applications***".
    - Intel manufactures dual and quad core versions; number of cores will likely increase to eight or even sixteen.

A Historical Background

Conceptual views of the 80486, Pentium Pro, Pentium II, Pentium III, Pentium 4, and Core2 microprocessors.

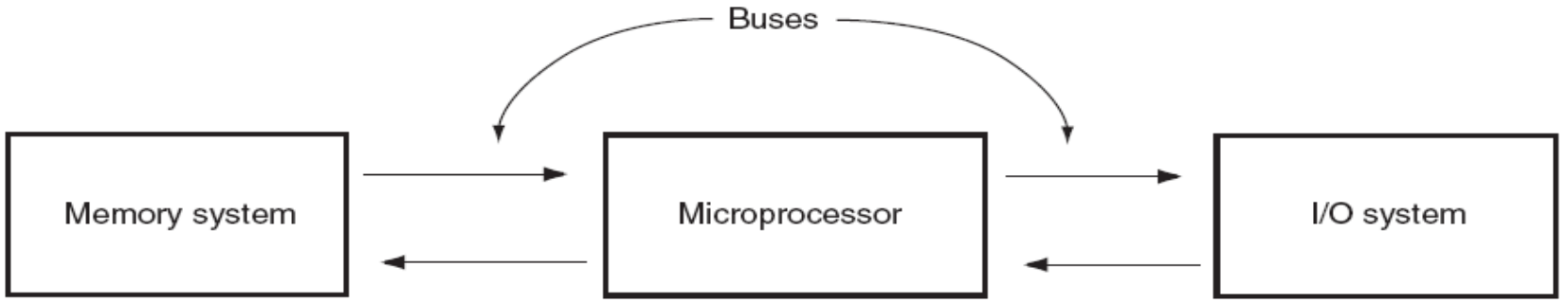


Outlines

1. Introduction
2. Important Definitions & Concepts
3. A Historical Background.
4. The Microprocessor-Based Computer System.
5. Number Systems.
6. Computer Data Formats.

The Microprocessor-Based Computer System

- The block diagram of a computer system:



Dynamic RAM (DRAM)
Static RAM (SRAM)
Cache
Read-only (ROM)
Flash memory
EEPROM
SDRAM
RAMBUS
DDR DRAM

8086
8088
80186
80188
80286
80386
80486
Pentium
Pentium Pro
Pentium II
Pentium III
Pentium 4
Core2

Printer
Serial communications
Floppy disk drive
Hard disk drive
Mouse
CD-ROM drive
Plotter
Keyboard
Monitor
Tape backup
Scanner
DVD

Outlines

1. Introduction
2. Important Definitions & Concepts
3. A Historical Background.
4. The Microprocessor-Based Computer System.
5. Number Systems.
6. Computer Data Formats.

Number Systems

Humans use decimal arithmetic (base 10) while computers use the binary (base 2) system.

Using of a microprocessor requires working knowledge of numbering systems.

- Decimal.
- Binary.
- Hexadecimal.

Number Systems

- Digits:
  - In ***base 10 (decimal)*** there are ***10 digits***:
    - 0, 1, 2, 3, 4, 5, 6, 7, 8 & 9.
  - In ***base 2 (binary)*** there are only ***two digits***:
    - 0 & 1.
    - Commonly referred to as ***bits***.
    - The binary system is used in computers because the two digits represent the two voltage levels *off & on.*
  - In ***base 16 (hexadecimal)*** there are ***10 digits and 6 letters***:
    - 0, 1, 2, 3, 4, 5, 6, 7, 8 , 9, A, B, C, D, E & F.

Outlines

1. Introduction
2. Important Definitions & Concepts
3. A Historical Background.
4. The Microprocessor-Based Computer System.
5. Number Systems.
6. Computer Data Formats.

Computer Data Formats

Because all information in the computer must be represented by 0s & 1s, binary patterns must be assigned to letters and other characters.

Common data formats:

- ASCII.
- Unicode.
- BCD.
- Signed and unsigned integers.
- Floating-point numbers (real numbers).

Other forms are available but are not commonly used.

Computer Data Formats

- ASCII data format:
  - (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) data format, represents ***alphanumeric characters*** in computer memory.
  - **Standard ASCII** code is a ***7-bit*** code.
  - Assigns binary patterns for:
    - Numbers 0 to 9.
    - All English alphabet letters, upper- and lower-case.
    - Also many control codes & punctuation marks.

Computer Data Formats

- ASCII data format:

TABLE 1–8 ASCII code.

	Second															
	X0	X1	X2	X3	X4	X5	X6	X7	X8	X9	XA	XB	XC	XD	XE	XF
First																
0X	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1X	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EMS	SUB	ESC	FS	GS	RS	US
2X	SP	!	“	#	$	%	&	’	(	)	*	+	,	–	.	/
3X	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4X	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5X	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6X	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7X	p	q	r	s	t	u	v	w	x	y	z	{	l	}	~	:::

Computer Data Formats

- ASCII data format:
  - (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) data format, represents ***alphanumeric characters*** in computer memory.
  - **Standard ASCII** code is a ***7-bit*** code.
  - The ***8th bit*** (MSB) in ***printing***:
    - If 0, for alphanumeric printing.
    - If 1, for graphics printing.

Computer Data Formats

- ASCII data format:
  - (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) data format, represents ***alphanumeric characters*** in computer memory.
  - **Standard ASCII** code is a ***7-bit*** code.
  - The ***8th bit*** (MSB) in ***personal computer***:
  - If 1, store some foreign letters and punctuation, Greek & mathematical characters, box-drawing & other special characters.
  - Called **Extended ASCII** characters.

Computer Data Formats

- ASCII data format:

TABLE 1–9 Extended ASCII code, as printed by the IBM ProPrinter.

First \ Second	X0	X1	X2	X3	X4	X5	X6	X7	X8	X9	XA	XB	XC	XD	XE	XF
0X		☺	☻	♥	♦	♣	♠	•	◘	○	◙	♂	♀	♪	♫	☼
1X	►	◄	↕	‼	¶	§	▬	↨	↑	↓	→	←	∟	↔	▲	▼
8X	Ç	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	Ä	Å
9X	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	¢	£	¥	₧	ƒ
AX	á	í	ó	ú	ñ	Ñ	ª	º	¿	⌐	¬	½	¼	¡	«	»
BX	░	▒	▓	│	┤	╡	╢	╖	╕	╣	║	╗	╝	╜	╛	┐
CX	└	┴	┬	├	─	┼	╞	╟	╚	╔	╩	╦	╠	═	╬	╧
DX	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	▀	▄	▌	▐	■
EX	α	β	Γ	π	Σ	σ	µ	γ	Φ	Θ	Ω	δ	∞	φ	∈	∩
FX	≡	±	≥	≤	⌠	⌡	÷	≈	°	∙	·	√	ⁿ	²	■	

Computer Data Formats

- Unicode data format:
  - Many Windows-based applications use the **Unicode** system to store alphanumeric data.
  - Stores each character as ***16-bit*** data.
  - Codes 0000H–00FFH are the same as standard ASCII code.
  - Remaining codes, 0100H–FFFFH, store all special characters from many character sets.
  - Allows software for Windows to be used in many countries around the world.

Computer Data Formats

- BCD data format:
  - (**B**inary-**C**oded **D**ecimal): digit from 0000_2 to 1001_2, (0 to 9 decimal).
  - Stored in two forms:
    - **Packed form:** stored as ***two digits per byte***.
      - Used for BCD addition and subtraction in the instruction set of the microprocessor.
    - **Unpacked form:** stored as ***one digit per byte***.
      - Returned from a keypad or keyboard (devices that perform a minimal amount of simple arithmetic).

Computer Data Formats

- BCD data format:
  - (**B**inary-**C**oded **D**ecimal): digit from 0000_2 to 1001_2, (0 to 9 decimal).
  - Stored in two forms:

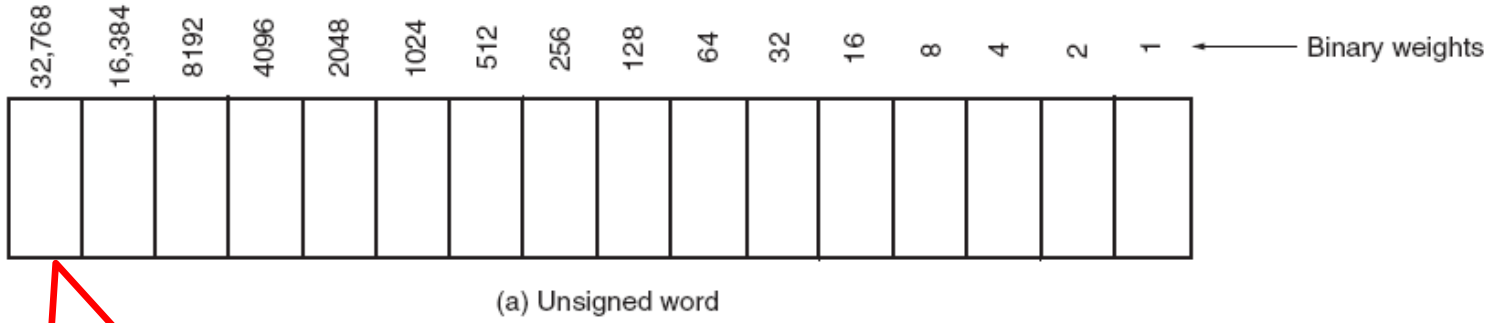
Decimal	*Packed*		*Unpacked*		
12	0001 0010		0000 0001	0000 0010	
623	0000 0110	0010 0011	0000 0110	0000 0010	0000 0011
910	0000 1001	0001 0000	0000 1001	0000 0001	0000 0000

Computer Data Formats

- Word-Sized data format:
  - A word (***16-bits***) is formed with **two bytes of data**.
  - The ***least significant byte*** always stored in the ***lowest-numbered memory location***.
    Most significant byte is stored in the highest.
  - This method of storing a number is called the ***little endian*** format.

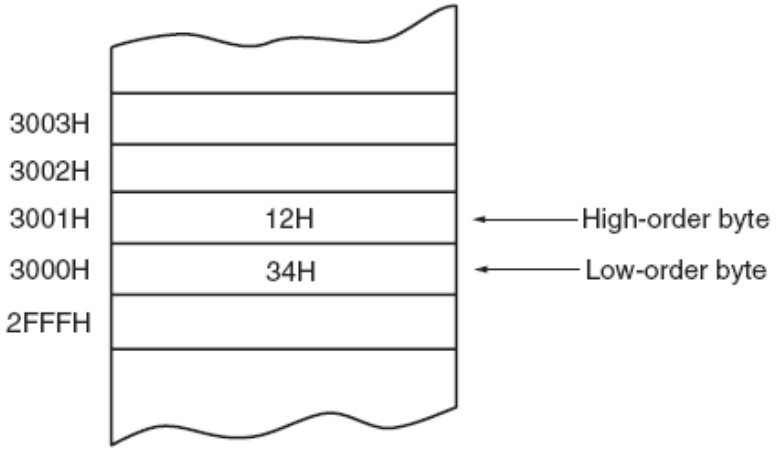
Computer Data Formats

- Word-Sized data format:
  - Example: figure 1-15:



(a) Unsigned word

The only difference between a signed and an unsigned word in the leftmost bit position.



(b) The contents of memory location 3000H and 3001H are the word 1234H.

Decimal System

A numbering system is a way for representing certain value in different ways.

Decimal system (base = 10): We have 10 symbols to represent values (0,1,2,3,4,5,6,7,8,9).

When finishing all the symbols in a digit, we make it zero and add one to the next digit.

$Value_{10}$
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

Decimal System

Decimal system (base = 10):

- Number of representable values in N digits: $M = 10^N$
- Range of values: $0 \rightarrow M - 1$
- Examples:

No. of digits (N)	No. of representable values (10^N)	Values range ($0 \rightarrow 10^N-1$)
1	10	$0 \rightarrow 9$
2	100	$0 \rightarrow 99$
3	1000	$0 \rightarrow 999$
4	10000	$0 \rightarrow 9999$

$Value_{10}$
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

Binary System

Binary system (base = 2): We have 2 symbols to represent values (0,1).

When finishing all the symbols in a digit (bit), we make it zero and add one to the next digit (bit).

We need too many bits for representing relatively small values:

$$(1000)_{10} = (11\ 1110\ 1000)_2$$

$$(1{,}000{,}000)_{10} =$$

$$(1111\ 0100\ 0010\ 0100\ 0000)_2$$

$Value_2$	$Value_{10}$
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15
10000	16

Hexadecimal System

Hexadecimal system (base = 16): We have 16 symbols to represent values (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F).

When finishing all the symbols in a digit, we make it zero and add one to the next digit.

Significantly less bits are required to represent the same values:

$$(1000)_{10} = (3E8)_{16}$$

$$(1,000,000)_{10} = (F4240)_{16}$$

$Value_2$	$Value_{10}$	$Value_{16}$
0	0	**0**
1	1	**1**
10	2	**2**
11	3	**3**
100	4	**4**
101	5	**5**
110	6	**6**
111	7	**7**
1000	8	**8**
1001	9	**9**
1010	10	**A**
1011	11	**B**
1100	12	**C**
1101	13	**D**
1110	14	**E**
1111	15	**F**
10000	16	**10**

Hexadecimal System

Hexadecimal system (base = 16):

- Number of representable values in N digits: $M = ?$
- Range of values: $0 \rightarrow ?$

$Value_2$	$Value_{10}$	$Value_{16}$
0	0	**0**
1	1	**1**
10	2	**2**
11	3	**3**
100	4	**4**
101	5	**5**
110	6	**6**
111	7	**7**
1000	8	**8**
1001	9	**9**
1010	10	**A**
1011	11	**B**
1100	12	**C**
1101	13	**D**
1110	14	**E**
1111	15	**F**
10000	16	**10**

Hexadecimal System

Hexadecimal system (base = 16):

- Number of representable values in N digits: $M = 16^N$
- Range of values: $0 \rightarrow M - 1$

$Value_2$	$Value_{10}$	$Value_{16}$
0	0	**0**
1	1	**1**
10	2	**2**
11	3	**3**
100	4	**4**
101	5	**5**
110	6	**6**
111	7	**7**
1000	8	**8**
1001	9	**9**
1010	10	**A**
1011	11	**B**
1100	12	**C**
1101	13	**D**
1110	14	**E**
1111	15	**F**
10000	16	**10**

Octal System

Octal system (base = 8): We have 8 symbols to represent values (0,1,2,3,4,5,6,7).

When finishing all the symbols in a digit, we make it zero and add one to the next digit.

Number of representable values in N digits: $M = 8^N$

Range of values: $0 \rightarrow M - 1$

$Value_2$	$Value_8$	$Value_{10}$	$Value_{16}$
0	**0**	0	0
1	**1**	1	1
10	**2**	2	2
11	**3**	3	3
100	**4**	4	4
101	**5**	5	5
110	**6**	6	6
111	**7**	7	7
1000	**10**	8	8
1001	**11**	9	9
1010	**12**	10	A
1011	**13**	11	B
1100	**14**	12	C
1101	**15**	13	D
1110	**16**	14	E
1111	**17**	15	F
10000	**20**	16	10

Other Systems

What about the base 5 system for example?

$Value_2$	$Value_5$	$Value_8$	$Value_{10}$	$Value_{16}$
0	?	0	0	0
1	?	1	1	1
10	?	2	2	2
11	?	3	3	3
100	?	4	4	4
101	?	5	5	5
110	?	6	6	6
111	?	7	7	7
1000	?	10	8	8
1001	?	11	9	9
1010	?	12	10	A
1011	?	13	11	B
1100	?	14	12	C
1101	?	15	13	D
1110	?	16	14	E
1111	?	17	15	F
10000	?	20	16	10

Other Systems

What about the base 5 system for example?

- We have 5 symbols to represent values (0,1,2,3,4).
- When finishing all the symbols in a digit, we make it zero and add one to the next digit.
- Number of representable values in N digits: $M = 5^N$

$Value_2$	$Value_5$	$Value_8$	$Value_{10}$	$Value_{16}$
0	0	0	0	0
1	1	1	1	1
10	2	2	2	2
11	3	3	3	3
100	4	4	4	4
101	10	5	5	5
110	11	6	6	6
111	12	7	7	7
1000	13	10	8	8
1001	14	11	9	9
1010	20	12	10	A
1011	21	13	11	B
1100	22	14	12	C
1101	23	15	13	D
1110	24	16	14	E
1111	30	17	15	F
10000	31	20	16	10

Other Systems

Any base-b system:

- There is no single symbol to represent the system base (R) or higher. System base is always represented as 10 in all numbering system.
- There will be R symbols to represent all values.
- Number of representable values in N digits: $M = R^N$.
- Range: $0 \rightarrow M - 1$

$Value_2$	$Value_5$	$Value_8$	$Value_{10}$	$Value_{16}$
0	0	0	0	0
1	1	1	1	1
10	2	2	2	2
11	3	3	3	3
100	4	4	4	4
101	10	5	5	5
110	11	6	6	6
111	12	7	7	7
1000	13	10	8	8
1001	14	11	9	9
1010	20	12	10	A
1011	21	13	11	B
1100	22	14	12	C
1101	23	15	13	D
1110	24	16	14	E
1111	30	17	15	F
10000	31	20	16	10

Value of Representation

To find the value of any representation in any numbering system we multiply each digit by its corresponding weight.

Decimal system: Example value of $(278)_{10}$:

Digits	2	7	8
Weights	10^2	10^1	10^0
Weights	100	10	1
Weighted digits	2×100	7×10	8×1
Value		278	

Value of Representation

To find the value of any representation in any numbering system we multiply each digit by its corresponding weight.

Decimal system: Example value of $(7129.45)_{10}$:

Digits	7	1	2	9 .	4	5
Weights	10^3	10^2	10^1	10^0	10^{-1}	10^{-2}
Weights	1000	100	10	1	0.1	0.01
Weighted	2×1000	7×100	2×10	9×1	4×0.1	5×0.01
Value	7129.45					

Converting to Decimal

To find the value of any representation in any numbering system we multiply each digit by its corresponding weight.

Binary system: Example value of $(1101)_2$:

Digits	1	1	0	1
Weights	2^3	2^2	2^1	2^0
Weights	8	4	2	1
Weighted digits	1×8	1×4	0×2	1×1
Value	$(13)_{10}$			

Converting to Decimal

To find the value of any representation in any numbering system we multiply each digit by its corresponding weight.

Octal system: Example value of $(263)_8$:

Digits	2	6	3
Weights	8^2	8^1	8^0
Weights	64	8	1
Weighted digits	2×64	6×8	3×1
Value	$(179)_{10}$		

Converting to Decimal

To find the value of any representation in any numbering system we multiply each digit by its corresponding weight.

Hexadecimal system: Example value of $(2A)_{16}$:

Digits	2	A
Weights	16^1	16^0
Weights	16	1
Weighted digits	2×16	A×1=?
Value	$(179)_{10}$	

A	10
B	11
C	12
D	13
E	14
F	15

Converting to Decimal

Generally, converting a number from any base R into decimal:

- Example: Given $N_R=(d_N d_{N-1}\ldots d_1 d_0 . d_{-1} d_{-2}\ldots d_{-P})_R$ convert it to base 10.

- N_{10}

d_N	d_{N-1}	…	d_1	d_0	d_{-1}	d_{-2}	…	d_{-P}
$d_n\times$ $\times R^N$	$\times R^{N-1}$		$\times R^1$	$\times R^0$	$\times R^{-1}$	$\times R^{-2}$		$\times R^{-P}$

$$+ d_{-1}\times R^{-1} + d_{-2}\times R^{-2} + \ldots + d_{-P}\times R^{-P}$$

$$N_{10} = \sum_{k=-P}^{N} d_k \times R^k$$

Value of Representation

This is how to convert a number from any numbering system into decimal.

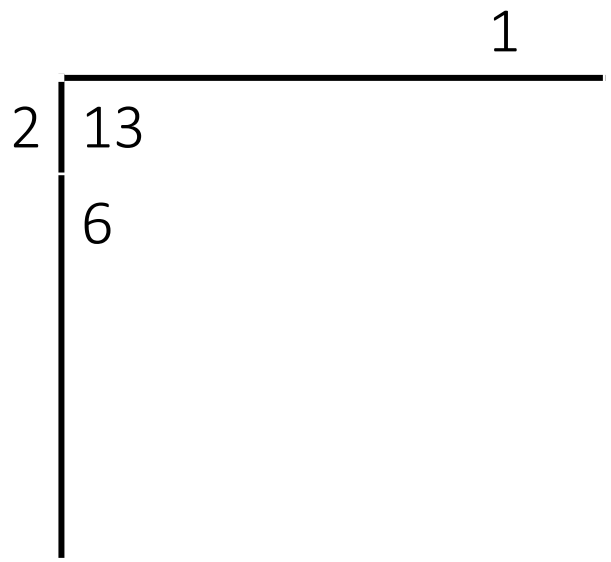
How to convert a decimal number into its representation in other systems?

Converting from Decimal

Converting a decimal **integer** to binary (base 2):

- Example: Given $(13)_{10}$
- By iterative division by 2 till reaching `0`.
- The representation is the reminder.

$$\frac{13}{2} = 6\frac{1}{2}$$

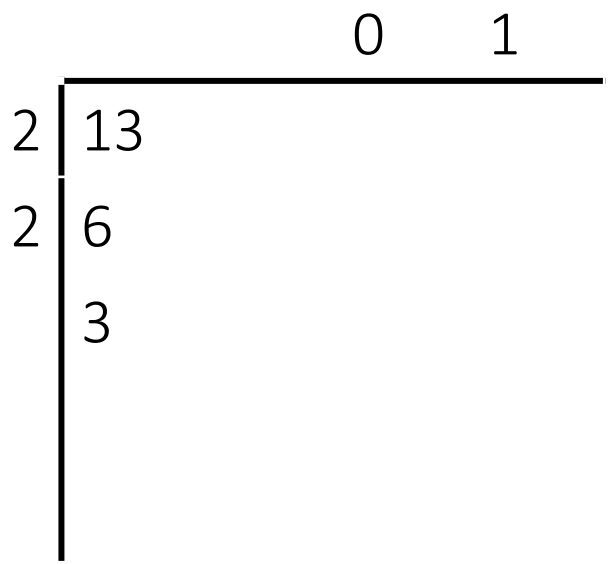


Converting from Decimal

Converting a decimal **integer** to binary (base 2):

- Example: Given $(13)_{10}$
- By iterative division by 2 till reaching 0.
- The representation is the reminder.

$$\frac{6}{2} = 3\frac{0}{2}$$



Converting from Decimal

Converting a decimal **integer** to binary (base 2):

- Example: Given $(13)_{10}$
- By iterative division by 2 till reaching 0.
- The representation is the reminder.

$$\frac{3}{2} = 1\frac{1}{2}$$

		1	0	1
2	13			
2	6			
2	3			
	1			

Converting from Decimal

Converting a decimal **integer** to binary (base 2):

- Example: Given $(13)_{10}$
- By iterative division by 2 till reaching **0**.
- The representation is the reminder.

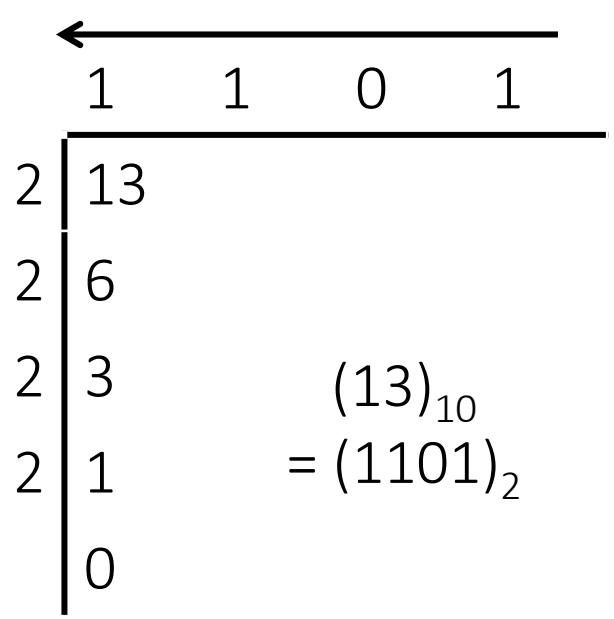
$$\frac{1}{2} = 0\frac{1}{2}$$

		1	1	0	1
2	13				
2	6				
2	3				
2	1				
	0				

Converting from Decimal

Converting a decimal **integer** to binary (base 2):

- Example: Given $(13)_{10}$
- By iterative division by 2 till reaching 0.
- The representation is the reminder.



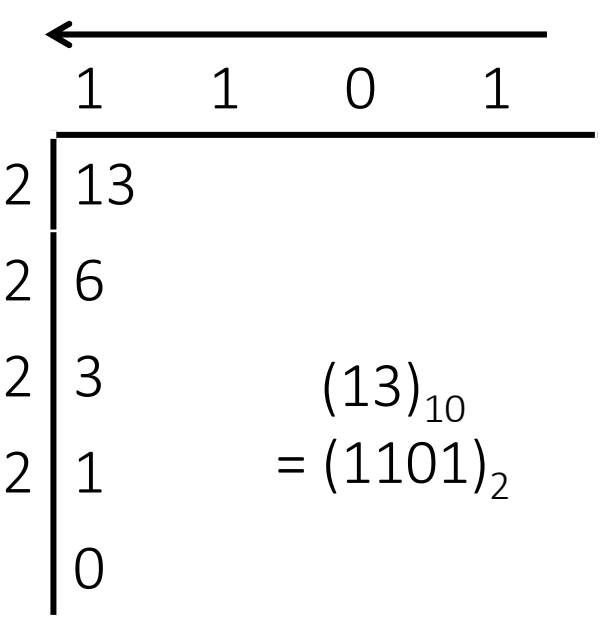
Converting from Decimal

Converting a decimal **integer** to binary (base 2):

- Example: Given $(13)_{10}$
- By iterative division by 2 till reaching 0.
- The representation is the reminder.

Converting a decimal **fraction** to binary (base 2):

- Example: Given $(0.375)_{10}$



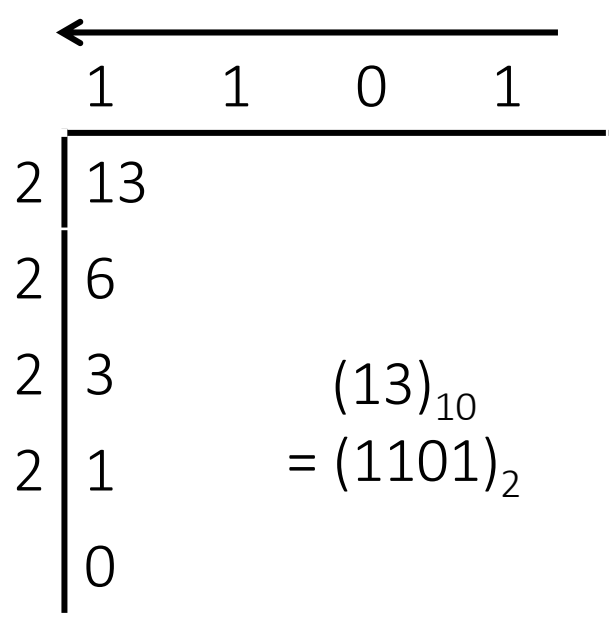
0.375

$(0.)_2$

Converting from Decimal

Converting a decimal **integer** to binary (base 2):

- Example: Given $(13)_{10}$
- By iterative division by 2 till reaching **0**.
- The representation is the reminder.



Converting a decimal **fraction** to binary (base 2):

- Example: Given $(0.375)_{10}$
- By iterative multiplication by 2 till reaching 0. Representation is the integer part.

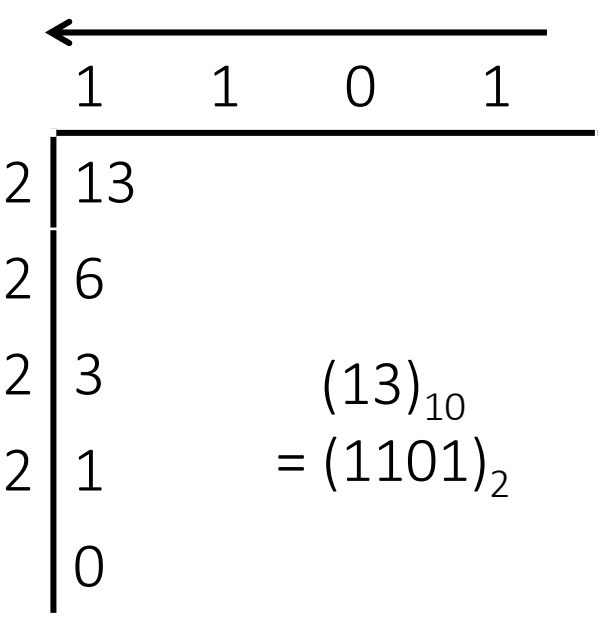
0.375 × 2 = **0**.75
0.75

$(0.0)_2$

Converting from Decimal

Converting a decimal **integer** to binary (base 2):

- Example: Given $(13)_{10}$
- By iterative division by 2 till reaching **0**.
- The representation is the reminder.



Converting a decimal **fraction** to binary (base 2):

- Example: Given $(0.375)_{10}$
- By iterative multiplication by 2 till reaching 0. Representation is the integer part.

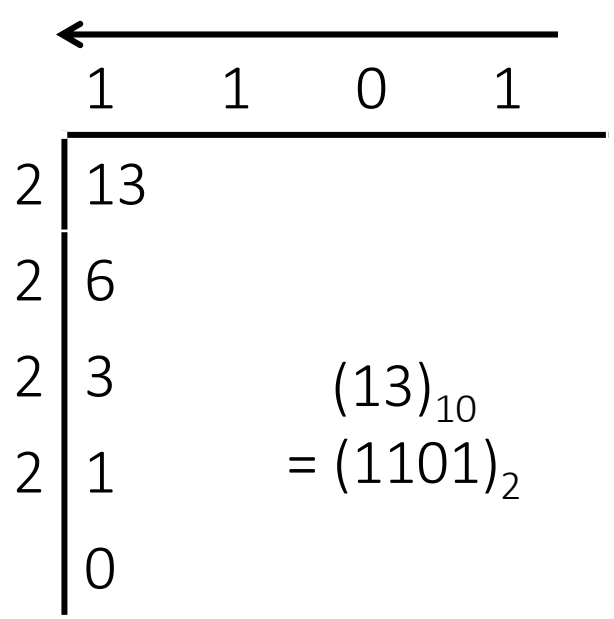
0.375 × 2 = 0.75
0.75 × 2 = **1**.5
0.5

$(0.01)_2$

Converting from Decimal

Converting a decimal **integer** to binary (base 2):

- Example: Given $(13)_{10}$
- By iterative division by 2 till reaching **0**.
- The representation is the reminder.



Converting a decimal **fraction** to binary (base 2):

- **Example: Given $(0.375)_{10}$**
- By iterative multiplication by 2 till reaching 0. Representation is the integer part.

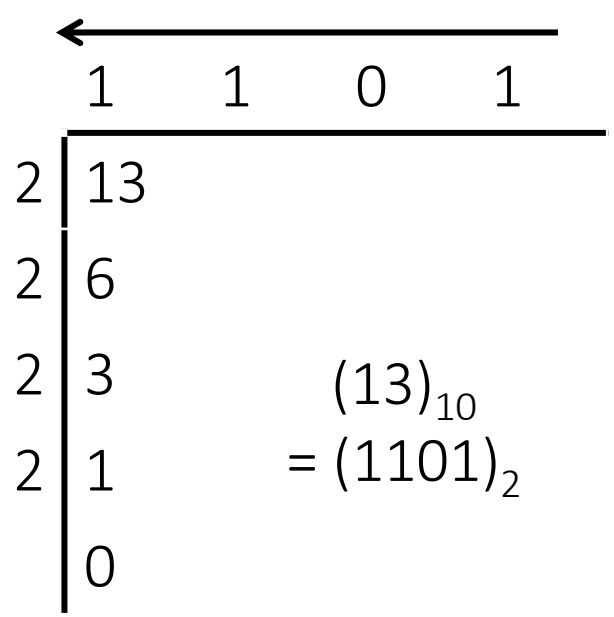
0.375 × 2 = 0.75
0.75 × 2 = 1.5
0.5 × 2 = **1**.0
0.0

$(0.011)_2$

Converting from Decimal

Converting a decimal **integer** to binary (base 2):

- Example: Given $(13)_{10}$
- By iterative division by 2 till reaching **0**.
- The representation is the reminder.



Converting a decimal **fraction** to binary (base 2):

- **Example: Given $(0.375)_{10}$**
- By iterative multiplication by 2 till reaching 0. Representation is the integer part.

0.375 × 2 = 0.75
0.75 × 2 = 1.5
0.5 × 2 = 1.0
0.0

$(0.375)_{10} = (0.011)_2$

Converting from Decimal

Converting a decimal **integer** to **octal** (base 8): HOW ??

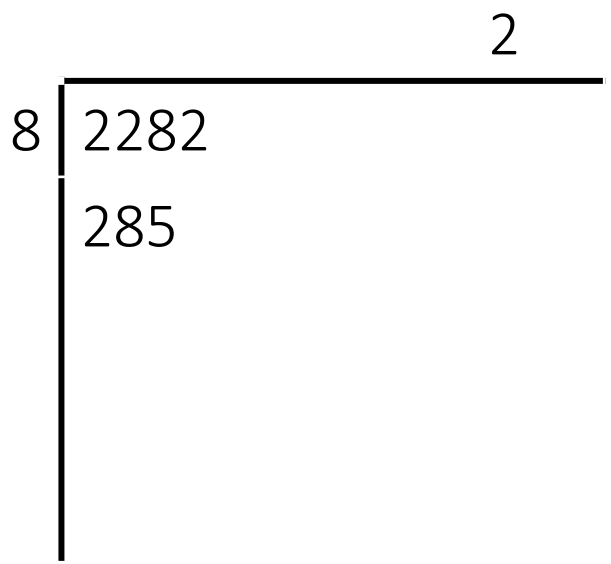
- Example: Given $(2282)_{10}$

Converting from Decimal

Converting a decimal **integer** to **octal** (base 8): HOW ??

- Example: Given $(2282)_{10}$
- By iterative division by 8 till reaching 0.
- The representation is the reminder.

$$\frac{2282}{8} = 285\frac{1}{4} = 285\frac{2}{8}$$



Converting from Decimal

Converting a decimal **integer** to **octal** (base 8): HOW ??

- Example: Given $(2282)_{10}$
- By iterative division by 8 till reaching 0.
- The representation is the reminder.

$$\frac{285}{8} = 35\frac{5}{8}$$

		5	2
8	2282		
8	285		
	35		

Converting from Decimal

Converting a decimal **integer** to **octal** (base 8): HOW ??

- Example: Given $(2282)_{10}$
- By iterative division by 8 till reaching 0.
- The representation is the reminder.

$$\frac{35}{8} = 4\frac{3}{8}$$

		3	5	2
8	2282			
8	285			
8	35			
	4			

Converting from Decimal

Converting a decimal **integer** to **octal** (base 8): HOW ??

- Example: Given $(2282)_{10}$
- By iterative division by 8 till reaching 0.
- The representation is the reminder.

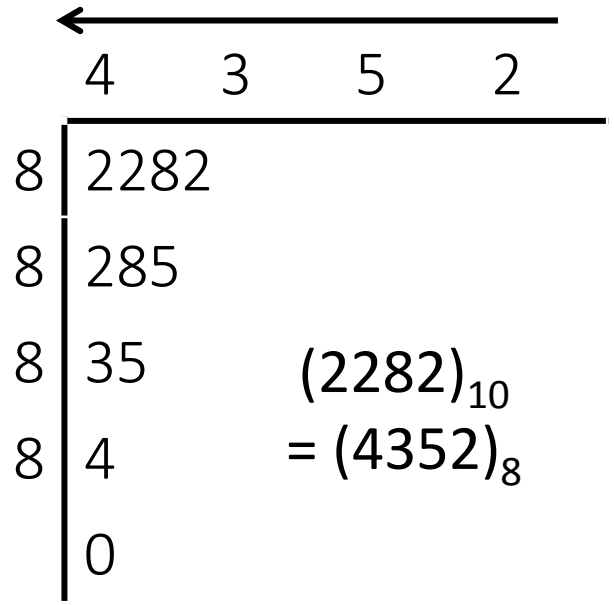
$$\frac{4}{8} = 0\frac{4}{8}$$

	4	3	5	2
8	2282			
8	285			
8	35			
8	4			
	0			

Converting from Decimal

Converting a decimal **integer** to **octal** (base 8): HOW ??

- Example: Given $(2282)_{10}$
- By iterative division by 8 till reaching 0.
- The representation is the reminder.



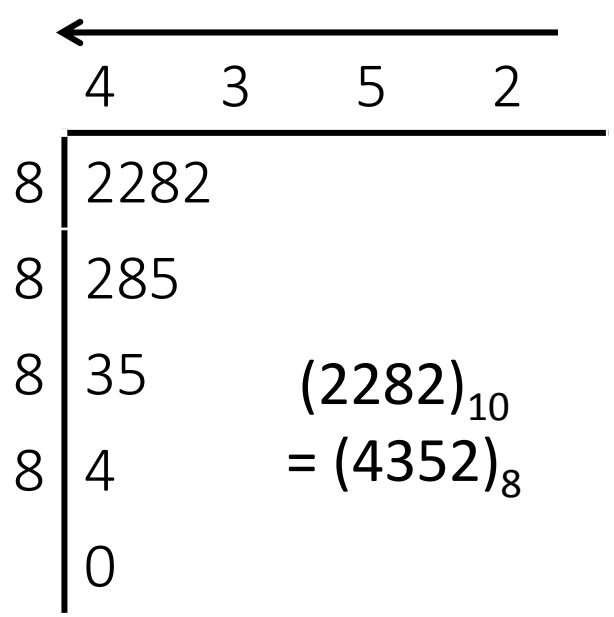
Converting a decimal **fraction** to **octal** (base 8): HOW ??

- Example: Given $(0.75)_{10}$

Converting from Decimal

Converting a decimal **integer** to **octal** (base 8): HOW ??

- Example: Given $(2282)_{10}$
- By iterative division by 8 till reaching 0.
- The representation is the reminder.



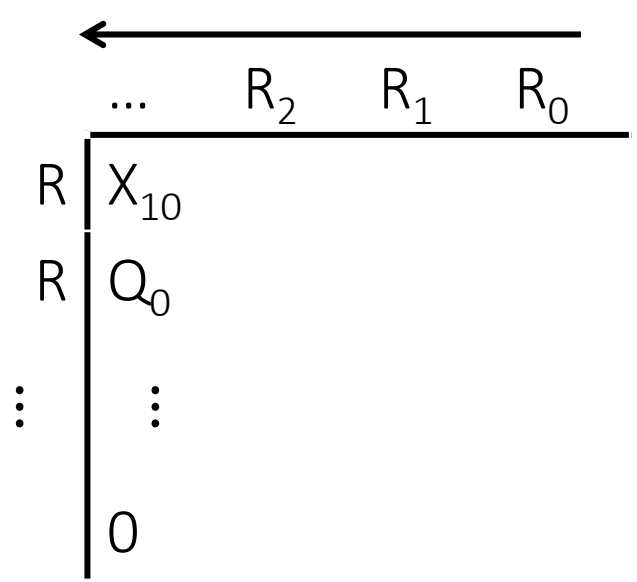
Converting a decimal **fraction** to **octal** (base 8): HOW ??

- Example: Given $(0.75)_{10}$
- By iterative multiplication by 8 till reaching 0. Representation is the integer part.

0.75 × 8 = **6**.0
0.0

$(0.75)_{10} = (0.6)_8$

Converting from Decimal



Generally, converting a decimal integer into base-b system:

- By iterative division by b till reaching 0.
- Given X_{10}

$$\frac{X_{10}}{b} = Q_0 \frac{R_0}{b}$$

$$\frac{Q_0}{b} = Q_1 \frac{R_1}{b}$$

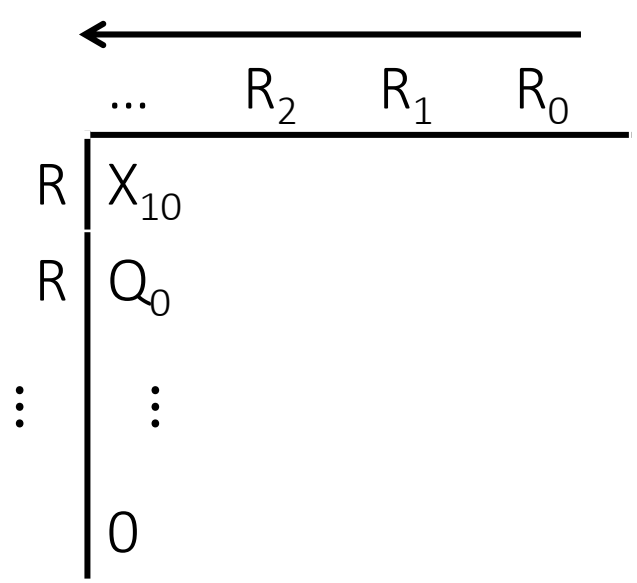
$$\dots$$

- $X_R = (\dots R_2 R_1 R_0)_b$

Converting from Decimal

Generally, converting a decimal fraction into base-b system:

- By iterative multiplication by b till reaching 0.
- Given X_{10}
- $X_R = (0.i_1 i_2 i_3 \ldots)_b$



$X_{10} \quad \times b = i_1.f_1$

$0.f_1 \quad \times b = i_2.f_2$

...

0.0

Converting from Decimal

Shortcut method for decimal to binary conversion:

- Put ones into the bits corresponding to composing weights.
- Example: Convert the numbers 9, 21, 12.25 into binary.

Weights	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}
Weights	32	16	8	4	2	1	0.5	0.25
9 =	0	0	1	0	0	1 .	0	0
21 =	0	1	0	1	0	1 .	0	0
12.25 =	0	0	1	1	0	0 .	0	1

Converting from Decimal

Shortcut method for decimal to binary conversion:

- Put ones into the bits corresponding to composing weights.
- Example: Convert the numbers 9, 21, 12.25 into binary.

$$(9)_{10} = (1001)_2$$

$$(21)_{10} = (10101)_2$$

$$(12.25)_{10} = (1100.01)_2$$

9 =	0	0	1	0	0	1	. 0	0
21 =	0	1	0	1	0	1	. 0	0
12.25 =	0	0	1	1	0	0	. 0	1

Converting from Decimal

Shortcut method for decimal to binary conversion:

- Put ones into the bits corresponding to composing weights.
- Example: Convert the numbers 9, 21, 12.25 into binary.

$$(9)_{10} = (1001)_2$$

$$(21)_{10} = (10101)_2$$

$$(12.25)_{10} = (1100.01)_2$$

- This can be done for any numbering system but consider multiplication of each weight.

Binary, Octal & Hexadecimal

Shortcut method: From binary to any other power-of-2 system (4,8,16):

- Given a number in binary and it is required to convert it into octal (base 8).
- Starting from the point, divide it into groups of $\log_2 8 = 3$ bits. Add zeros to the left in integer and right in fraction if needed.
- Convert each group as binary into octal digit.

$(10110.01)_2$

↓

010 110 . 01**0**

↓

$(26.2)_8$

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

Binary, Octal & Hexadecimal

Shortcut method: From binary to any other power-of-2 system (4,8,16):

- Given a number in binary and it is required to convert it into hexadecimal (base 16). HOW?
- Starting from the point, divide it into groups of $\log_2 16 = 4$ bits. Add zeros to the left in integer and right in fraction if needed.
- Convert each group as binary into hexadecimal digit.

$(1011110)_2$

↓

0101 1110

↓

$(5E)_{16}$

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

Binary, Octal & Hexadecimal

Shortcut method: From any power-of-2 system (4,8,16) to binary:

- Given a number in octal (base 8) and it is required to convert it into binary. HOW?
- Starting from the point, convert each digit into group of $\log_2 16 = 4$ bits. Add zeros to the left in integer and right in fraction if needed.

$(23.2)_8$

↓

010 011 . 01**0**

↓

$(10011.01)_2$

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

Binary, Octal & Hexadecimal

Shortcut method: From any power-of-2 system (4,8,16) to binary:

- Given a number in hexadecimal (base 16) and it is required to convert it into binary. HOW?
- Starting from the point, convert each digit into group of $\log_2 8 = 3$ bits. Add zeros to the left in integer and right in fraction if needed.

$(A5)_{16}$

↓

1010 **0**101

↓

$(10100101)_2$

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

Binary, Octal & Hexadecimal

How to convert between octal and hexadecimal?

- First convert to binary then to the required system.

$(26.2)_8$

↓

$(010\ 110\ .\ 010)_2$

↓

$(\mathbf{00}01\ 0110\ .\ 010\mathbf{0})_2$

↓

$(16.4)_{16}$

- Or

$(16.4)_{16}$

↓

$(0001\ 0110\ .\ 0100)_2$

↓

$(010\ 110\ .\ 010)_2$

↓

$(26.2)_8$

Thanks

