Lecture 15 Micro Processors

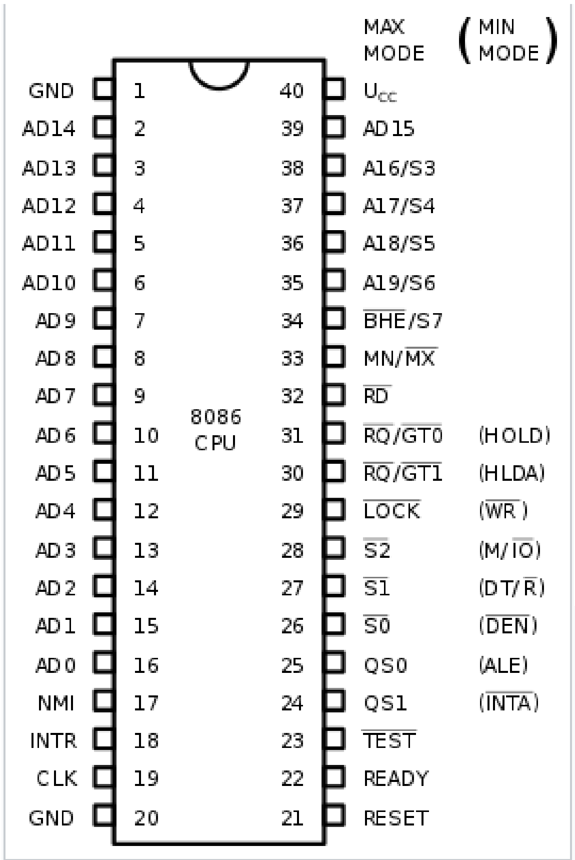
:Packaged Chips for Logical and Arithmetic Calculation

Micro Processor	vs	Logical/Arithmetic Gates
Firm-ware(or softwired) based on Packaged Chip		Hardwired configuration
Modification by rewriting new firm-ware		Modification by rewiring
Software based on 8086/8088/80X86 (e.g.) (Few 1000 gates typically)		In-house design using Logic/Arithmetic Gates

LSI: Large Scale Integration ≒ 1000 gates

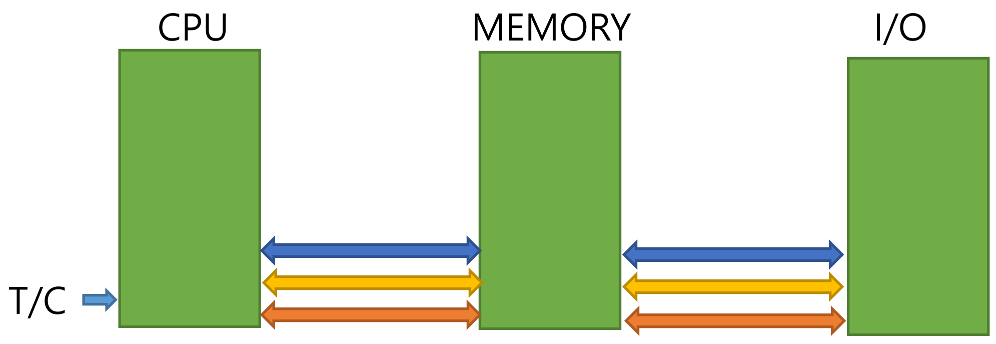
VLSI: Very Large Scale Integration ≒ Few thousands gate that most CPU belong to.

8086 Micro Processor


MAX MODE
(MIN MODE)
8086 CPU
GND 1 — 40 U_{CC}
AD14 2 — 39 AD15
AD13 3 — 38 A16/S3
AD12 4 — 37 A17/S4
AD11 5 — 36 A18/S5
AD10 6 — 35 A19/S6
AD9 7 — 34 $\overline{\text{BHE}}$/S7
AD8 8 — 33 MN/$\overline{\text{MX}}$
AD7 9 — 32 $\overline{\text{RD}}$
AD6 10 — 31 $\overline{\text{RQ}}/\overline{\text{GT0}}$ (HOLD)
AD5 11 — 30 $\overline{\text{RQ}}/\overline{\text{GT1}}$ (HLDA)
AD4 12 — 29 $\overline{\text{LOCK}}$ ($\overline{\text{WR}}$)
AD3 13 — 28 $\overline{\text{S2}}$ (M/$\overline{\text{IO}}$)
AD2 14 — 27 $\overline{\text{S1}}$ (DT/$\overline{\text{R}}$)
AD1 15 — 26 $\overline{\text{S0}}$ ($\overline{\text{DEN}}$)
AD0 16 — 25 QS0 (ALE)
NMI 17 — 24 QS1 ($\overline{\text{INTA}}$)
INTR 18 — 23 $\overline{\text{TEST}}$
CLK 19 — 22 READY
GND 20 — 21 RESET
The 8086 pin assignments in min and max mode

(source:en.wikipedia.org/wiki/Intel_8086)

<u>Overview of Micro Processor System</u>



Address Bus: ⟷ Data Bus: ⟷ Control Bus: ⟷

T/C=Timing Circuit

CPU (Central Processing Unit)

-To control the whole processes like brain

-To decode instructions such that 03C3 (=0000 0011 1101 0011) indicates ADD (or +), etc

-To execute the instructions such as arithmetic/logic computation, by

connecting/calling gates to/from internal registers, memory, I/O devices

Timing Circuit (or CLK)

To generate trains of time pulses, and

To synchronize the activity within the system

Memory

To store both the data and instructions, RAM (Random Access memory) and ROM (Read Only Memory)

I/O (Input/output) Device

-To communicate with the external world, and

-To store large quantity of information, and

-To make input/output such as Keyboard, Mouse, USB, ADC, DAC,

Hard Disc, Monitor

System Bus

-To connect CPU to memory and I/O (memory mapped I/O)

(1) Address Bus (A0-A19 for 8086)

To transfer address (20 bits for 8086) to and from Memory, I/O

2^{20} Locations=10^{6} Locations= 1 Mega Byte Memory locations accessible

(2) Data Bus (D0-D15 for 8086)

:To transfer data (16 bits for 8086) to and from memory, I/O

Number of data lines= data width=16 for 8086 ∴16 bit processor

And two bus cycle is needed to process 32 bits data

(3) Control Bus

To issue a control command such as Read, Write, Acknowledge, and Interrupt Request

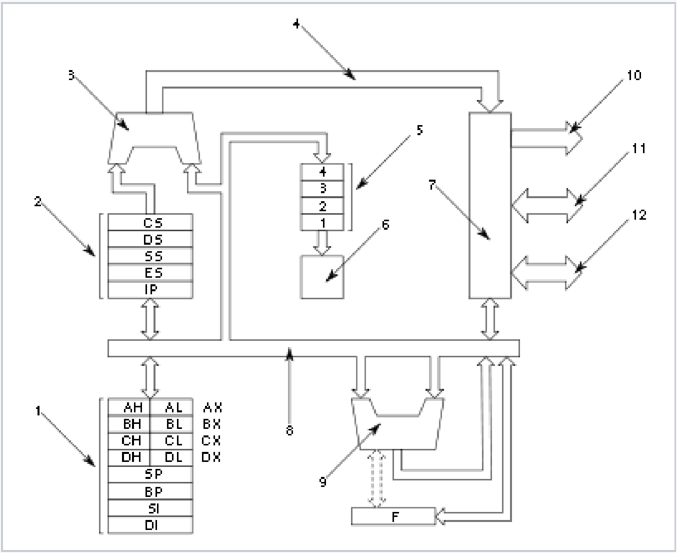
Interface

Circuit to connect the Bus to a device, such as memory and I/O

Memory or I/O Interface

=Address Decoding + Data Buffering + Read/Write

8086 Micro Processor Structure



Simplified block diagram over Intel 8088 (a variant of 8086); 1=main registers; 2=segment registers and IP; 3=address adder; 4=internal address bus; 5=instruction queue; 6=control unit (very simplified!); 7=bus interface; 8=internal databus; 9=ALU; 10/11/12=external address/data/control bus.

(source:en.wikipedia.org/wiki/Intel_8086)

ALU (Arithmetic Logic Unit)

-Arithmetic and Logic Computation

-I/O control + I/O handling

IR (Instruction Register)

-To hold the current instruction from the instruction queue

IP (Instruction Pointer)

-To hold the address of the next instruction

Working Registers: AX, BX, CX, DX

-To store temporary information for computing and addressing process

-To store arithmetic/logic operands, results

Each working register of 16 bits consists of higher 8bit and lower 8bit

such as AX=AH+AL, BX=BH+BL, CX=CH+CL, DX=DH+DL, thus 16bit processing and 8bits processing are both possible.

Also for special functions such as;

AX: I/O data handling as accumulator

DX: I/O address handling

BX: Base register for address calculation

CX: Implied counter

Segment Registers: CS, DS, ES, SS

Register to hold the address of each segments within the memory map

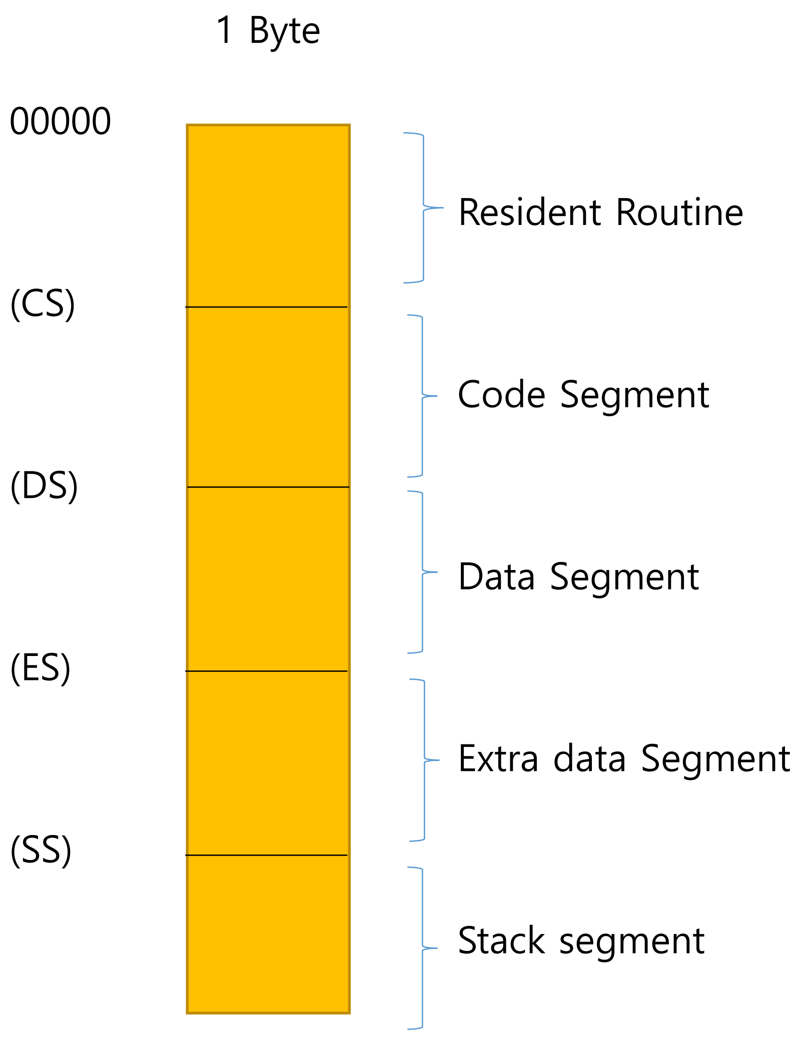
CS=Code Segment register to hold the address of code (or instruction) segment

DS=Data Segment register to hold the address of data segment

ES=Extra Segment register to hold the address of extra data segment

SS=Stack Segment register to hold the address of the stack segment

Memory Map within a Microprocessor system (8086)



Physical Address=(Segment Register)X10_{16} + Effective Address

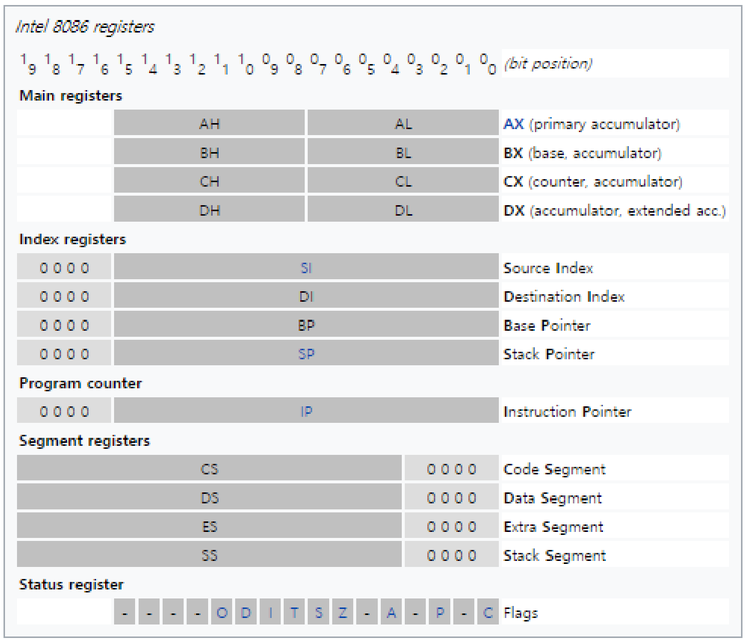
where Effective Address (EA) is the relative distance within the segment

Index registers: Registers for index/base/stack pointing

SI(Source Index), DI(Destination Index), BP(Base Pointer), SP(Stack Pointer)

SP (Stack Pointer)

-To store the address of stack location, where stack is the memory space of temporary contents of working registers during the subroutine calling



Intel 8086 registers

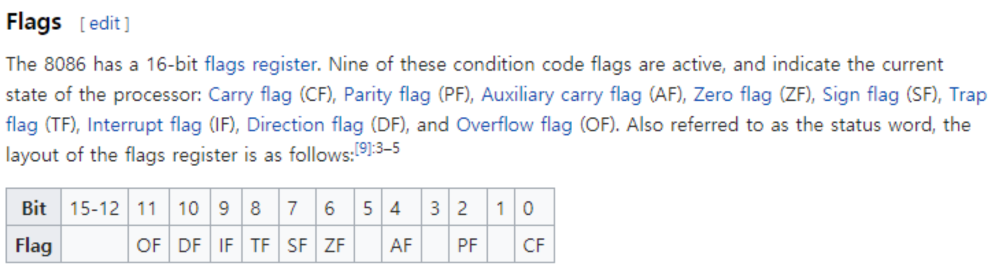
$^1_9$ $^1_8$ $^1_7$ $^1_6$ $^1_5$ $^1_4$ $^1_3$ $^1_2$ $^1_1$ $^1_0$ $^0_9$ $^0_8$ $^0_7$ $^0_6$ $^0_5$ $^0_4$ $^0_3$ $^0_2$ $^0_1$ $^0_0$ *(bit position)*

Main registers			
	AH	AL	**AX** (primary accumulator)
	BH	BL	**BX** (base, accumulator)
	CH	CL	**CX** (counter, accumulator)
	DH	DL	**DX** (accumulator, extended acc.)
Index registers			
0 0 0 0	SI		Source Index
0 0 0 0	DI		Destination Index
0 0 0 0	BP		Base Pointer
0 0 0 0	SP		Stack Pointer
Program counter			
0 0 0 0	IP		Instruction Pointer
Segment registers			
CS		0 0 0 0	Code Segment
DS		0 0 0 0	Data Segment
ES		0 0 0 0	Extra Segment
SS		0 0 0 0	Stack Segment
Status register			
	- - - - O D I T S Z - A - P - C		Flags

(source:en.wikipedia.org/wiki/Intel_8086)

PSW (Process Status Word)

-To store the current status of the processor such as zero, parity, sign, overflow, carry, interrupt, etc. It is called as 'Flags'



Flags [edit]

The 8086 has a 16-bit flags register. Nine of these condition code flags are active, and indicate the current state of the processor: Carry flag (CF), Parity flag (PF), Auxiliary carry flag (AF), Zero flag (ZF), Sign flag (SF), Trap flag (TF), Interrupt flag (IF), Direction flag (DF), and Overflow flag (OF). Also referred to as the status word, the layout of the flags register is as follows:[9]:3–5

Bit	15-12	11	10	9	8	7	6	5	4	3	2	1	0
Flag		OF	DF	IF	TF	SF	ZF		AF		PF		CF

(source:en.wikipedia.org/wiki/Intel_8086)

CF=1 if previous computation results in 'Carry'

PF=1 if previous computation results in 'Parity'

(or Even number of '1' in lower 8bits after computation)

ZF=1 if previous computation results in zero

SF=1 if previous computation results in negative sign

Operation Sequence of Micro processor

1. Fetch the next instruction from the address indicated by IP
2. Put it into the IR and decode it, while IP is incremented to point to the next instruction
3. Execute the instruction
4. Repeat 1 through 3

Data Formats in Micro processor

1. Numerical Data

(1) Unsigned binary

$N=b_{n-1}b_{n-2}....b_1b_0=b_{n-1}2^{n-1}+b_{n-2}2^{n-2}+..+b_12^1+b_02^0$

To express in Hexadecimal;

0000 0001 0010 0011 0100 0101 0110 0111 -> 0 1 2 3 4 5 6 7

1000 1001 1010 1011 1100 1101 1110 1111 -> 8 9 A B C D E F

(2) Signed binary

:Negative number as 2's complement

$-N=2^n-N\equiv\underline{N}+1$, where $\underline{N}$ is the negate of N in binary format

Ex)-45=? for signed binary

$45=32+8+4+1=00101101_{(2)}$ thus $\underline{45}=11010010$

$\therefore -45=\underline{N}+1=11010011_{(2)}=D3_{(16)}$

(3) Packed BCD (Binary Coded Decimal)

Each decimal digit is encoded into 4 bit pattern

3509 in BCD format= 0011010100001001

3509 in Decimal=2048+1024+256+128+32+16+4+1

=0000110110110101

(4) Fixed Point Format

$.F=.b_{-1}\ b_{-2}\ldots b_{-n}=b_{-1}2^{-1}+b_{-2}2^{-2}+..+b_{-n}2^{-n}$

0.59375=?

$0.59375x2=1.1875=\underline{1}+.1875$, $0.1875x2=0.3750=\underline{0}+.3750$

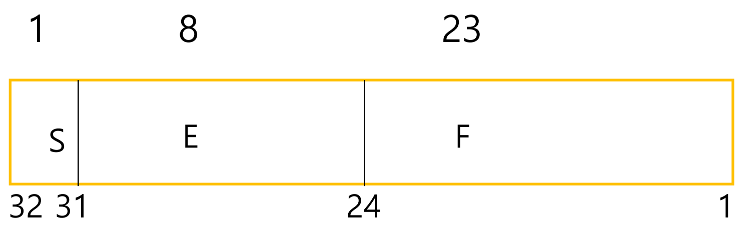
$0.3750x2=0.750=\underline{0}+.750$, $0.750x2=1.5=\underline{1}+.5$, $0.5x2=1.0=\underline{1}$

Thus $0.59375=0.10011_{(2)}$

(5) Floating point format (Single Precision)

To express a real number as 4 byte expression as,

Real Number=$(-1)^S$ x 2^{E-127} x 1.F



Range of real number : $\pm 1\text{x}10^{-38}$ to $\pm 3\text{x}10^{38}$

Ex1) 313.125=313+0.125=(256+32+16+8+1)+(2^{-3})

=$100111001.001_{(2)}$=1.00111001001x2^8

S=0, E-127=8 ∴E=135=128+4+2+1=$10000111_{(2)}$

F=00111001001

Thus 313.125=0100 0011 1001 1100 1001

=439C9000= Floating Format (Single Precision)

Ex2) Single Precision C25A8000 ?

1100 0010 0101 1010 1000 0000 0000 0000

=1 10000100 10110101000 0000 0000 0000

∴S=1, E=128+4=132 thus E-127=5, F=.10110101

∴Real Number=-1.10110101x2^5= $-110110.101_{(2)}$

=-(32+16+4+2).(0.5+0.125)=-54.625

2. Non-numerical Data

:To express the non-numerical data as

ASCII code (American Standard Code for Information Interchange)

00~1F : Non printable control characters

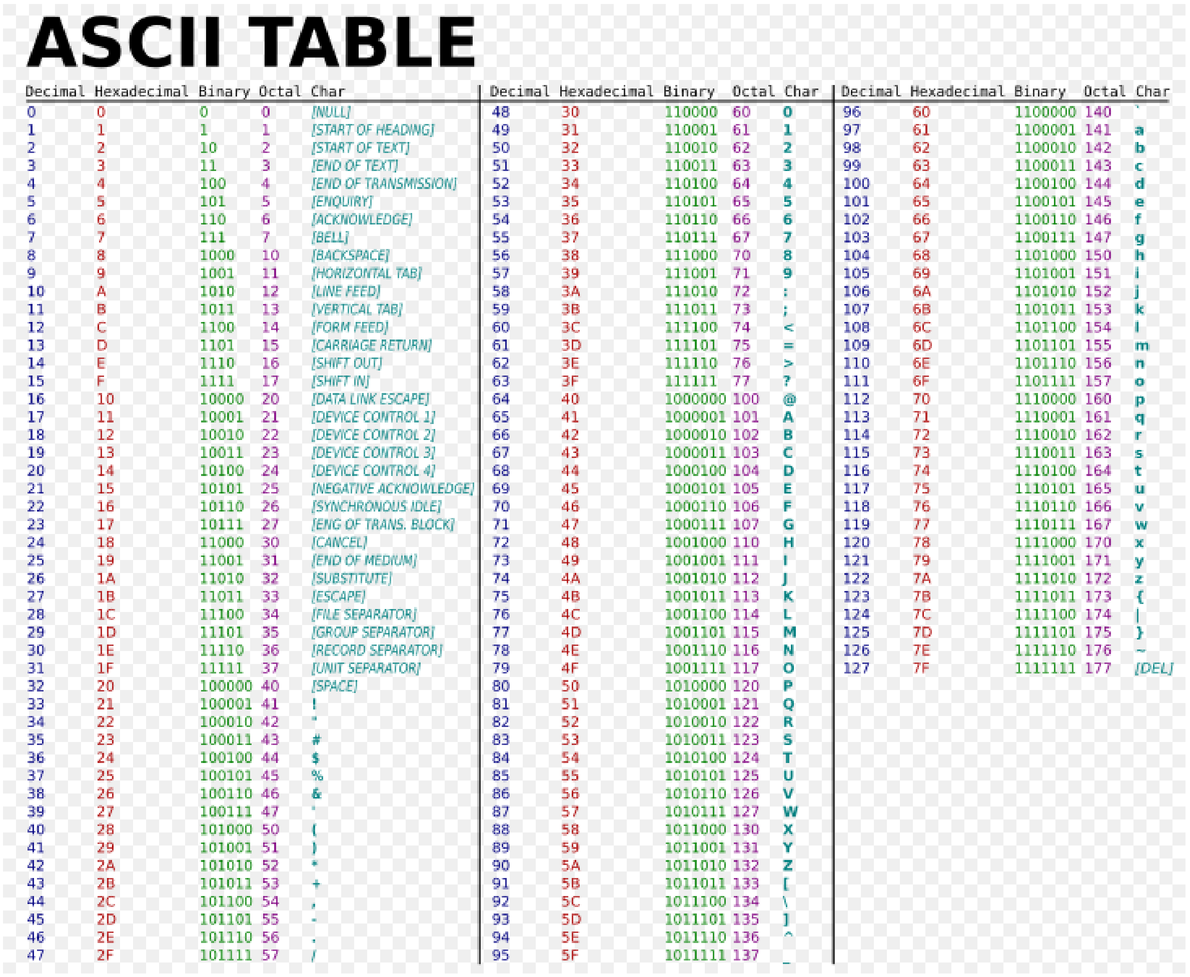
20~7F : Printable characters

80~FF : Extended ASCII code for local language customization

Ex) ‘1’=31H=00110001B

’A’=41H=01000001B

‘a’=61H=01100001B



ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]
1	1	1	1	[START OF HEADING]
2	2	10	2	[START OF TEXT]
3	3	11	3	[END OF TEXT]
4	4	100	4	[END OF TRANSMISSION]
5	5	101	5	[ENQUIRY]
6	6	110	6	[ACKNOWLEDGE]
7	7	111	7	[BELL]
8	8	1000	10	[BACKSPACE]
9	9	1001	11	[HORIZONTAL TAB]
10	A	1010	12	[LINE FEED]
11	B	1011	13	[VERTICAL TAB]
12	C	1100	14	[FORM FEED]
13	D	1101	15	[CARRIAGE RETURN]
14	E	1110	16	[SHIFT OUT]
15	F	1111	17	[SHIFT IN]
16	10	10000	20	[DATA LINK ESCAPE]
17	11	10001	21	[DEVICE CONTROL 1]
18	12	10010	22	[DEVICE CONTROL 2]
19	13	10011	23	[DEVICE CONTROL 3]
20	14	10100	24	[DEVICE CONTROL 4]
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]
22	16	10110	26	[SYNCHRONOUS IDLE]
23	17	10111	27	[ENG OF TRANS. BLOCK]
24	18	11000	30	[CANCEL]
25	19	11001	31	[END OF MEDIUM]
26	1A	11010	32	[SUBSTITUTE]
27	1B	11011	33	[ESCAPE]
28	1C	11100	34	[FILE SEPARATOR]
29	1D	11101	35	[GROUP SEPARATOR]
30	1E	11110	36	[RECORD SEPARATOR]
31	1F	11111	37	[UNIT SEPARATOR]
32	20	100000	40	[SPACE]
33	21	100001	41	!
34	22	100010	42	"
35	23	100011	43	#
36	24	100100	44	$
37	25	100101	45	%
38	26	100110	46	&
39	27	100111	47	'
40	28	101000	50	(
41	29	101001	51	)
42	2A	101010	52	*
43	2B	101011	53	+
44	2C	101100	54	,
45	2D	101101	55	-
46	2E	101110	56	.
47	2F	101111	57	/
48	30	110000	60	0
49	31	110001	61	1
50	32	110010	62	2
51	33	110011	63	3
52	34	110100	64	4
53	35	110101	65	5
54	36	110110	66	6
55	37	110111	67	7
56	38	111000	70	8
57	39	111001	71	9
58	3A	111010	72	:
59	3B	111011	73	;
60	3C	111100	74	<
61	3D	111101	75	=
62	3E	111110	76	>
63	3F	111111	77	?
64	40	1000000	100	@
65	41	1000001	101	A
66	42	1000010	102	B
67	43	1000011	103	C
68	44	1000100	104	D
69	45	1000101	105	E
70	46	1000110	106	F
71	47	1000111	107	G
72	48	1001000	110	H
73	49	1001001	111	I
74	4A	1001010	112	J
75	4B	1001011	113	K
76	4C	1001100	114	L
77	4D	1001101	115	M
78	4E	1001110	116	N
79	4F	1001111	117	O
80	50	1010000	120	P
81	51	1010001	121	Q
82	52	1010010	122	R
83	53	1010011	123	S
84	54	1010100	124	T
85	55	1010101	125	U
86	56	1010110	126	V
87	57	1010111	127	W
88	58	1011000	130	X
89	59	1011001	131	Y
90	5A	1011010	132	Z
91	5B	1011011	133	[
92	5C	1011100	134	\
93	5D	1011101	135	]
94	5E	1011110	136	^
95	5F	1011111	137	_
96	60	1100000	140	`
97	61	1100001	141	a
98	62	1100010	142	b
99	63	1100011	143	c
100	64	1100100	144	d
101	65	1100101	145	e
102	66	1100110	146	f
103	67	1100111	147	g
104	68	1101000	150	h
105	69	1101001	151	i
106	6A	1101010	152	j
107	6B	1101011	153	k
108	6C	1101100	154	l
109	6D	1101101	155	m
110	6E	1101110	156	n
111	6F	1101111	157	o
112	70	1110000	160	p
113	71	1110001	161	q
114	72	1110010	162	r
115	73	1110011	163	s
116	74	1110100	164	t
117	75	1110101	165	u
118	76	1110110	166	v
119	77	1110111	167	w
120	78	1111000	170	x
121	79	1111001	171	y
122	7A	1111010	172	z
123	7B	1111011	173	{
124	7C	1111100	174	|
125	7D	1111101	175	}
126	7E	1111110	176	~
127	7F	1111111	177	[DEL]

(source:commons.wikipedia.org/wiki/File:ASCII-Table.svg)