Nama : Reylian Prealdream Anareka
NIM : L20080087
Kelas : B

ASCII

Kode Standar Amerika untuk Pertukaran Informasi atau American Standard Code for Information Interchange (ASCII) merupakan suatu standar internasional dalam kode huruf dan simbol seperti Hex dan Unicode tetapi ASCII lebih bersifat universal, contohnya 124 adalah untuk karakter "|". Ia selalu digunakan oleh komputer dan alat komunikasi lain untuk menunjukkan teks. Kode ASCII sebenarnya memiliki komposisi bilangan biner sebanyak 7 bit. Namun, ASCII disimpan sebagai sandi 8 bit dengan menambakan satu angka 0 sebagai bit significant paling tinggi. Bit tambahan ini sering digunakan untuk uji paritas. Karakter control pada ASCII dibedakan menjadi 5 kelompok sesuai dengan penggunaan yaitu berturut-turut meliputi logical communication, Device control, Information separator, Code extention, dan physical communication. Code ASCII ini banyak dijumpai pada papan ketik (keyboard) computer atau instrument-instrument digital.

Jumlah kode ASCII adalah 255 kode. Kode ASCII 0..127 merupakan kode ASCII untuk manipulasi teks; sedangkan kode ASCII 128..255 merupakan kode ASCII untuk manipulasi grafik. Kode ASCII sendiri dapat dikelompokkan lagi kedalam beberapa bagian:

- Kode yang tidak terlihat simbolnya seperti Kode 10(Line Feed), 13(Carriage Return), 8(Tab), 32(Space)
- Kode yang terlihat simbolnya seperti abjad (A..Z), numerik (0..9), karakter khusus (~!@#$%^&*()_+?:”{})
- Kode yang tidak ada di keyboard namun dapat ditampilkan. Kode ini umumnya untuk kode-kode grafik.

Dalam pengkodean kode ASCII memanfaatkan 8 bit. Pada saat ini kode ASCII telah tergantikan oleh kode UNICODE (Universal Code). UNICODE dalam pengkodeannya memanfaatkan 16 bit sehingga memungkinkan untuk menyimpan kode-kode lainnya seperti kode bahasa Jepang, Cina, Thailand dan sebagainya.

Pada papan keyboard, aktifkan numlock, tekan tombol ALT secara bersamaan dengan kode karakter maka akan dihasilkan karakter tertentu. Misalnya: ALT + 44 maka akan muncul karakter koma (,). Mengetahui kode-kode ASCII sangat bermanfaat misalnya untuk membuat karakter-karakter tertentu yang tidak ada di keyboard.

Tabel Karakter ASCII

DEC	OCT	HEX	BIN	Symbol	Description
0	000	00	00000000	NUL	Null char
1	001	01	00000001	SOH	Start of Heading
2	002	02	00000010	STX	Start of Text
3	003	03	00000011	ETX	End of Text
4	004	04	00000100	EOT	End of Transmission
5	005	05	00000101	ENQ	Enquiry
6	006	06	00000110	ACK	Acknowledgment
7	007	07	00000111	BEL	Bell
8	010	08	00001000	BS	Back Space
9	011	09	00001001	HT	Horizontal Tab
10	012	0A	00001010	LF	Line Feed
11	013	0B	00001011	VT	Vertical Tab
12	014	0C	00001100	FF	Form Feed

13	015	0D	00001101	CR	Carriage Return
14	016	0E	00001110	SO	Shift Out / X-On
15	017	0F	00001111	SI	Shift In / X-Off
16	020	10	00010000	DLE	Data Line Escape
17	021	11	00010001	DC1	Device Control 1 (oft. XON)
18	022	12	00010010	DC2	Device Control 2
19	023	13	00010011	DC3	Device Control 3 (oft. XOFF)
20	024	14	00010100	DC4	Device Control 4
21	025	15	00010101	NAK	Negative Acknowledgement
22	026	16	00010110	SYN	Synchronous Idle
23	027	17	00010111	ETB	End of Transmit Block
24	030	18	00011000	CAN	Cancel
25	031	19	00011001	EM	End of Medium
26	032	1A	00011010	SUB	Substitute
27	033	1B	00011011	ESC	Escape
28	034	1C	00011100	FS	File Separator
29	035	1D	00011101	GS	Group Separator
30	036	1E	00011110	RS	Record Separator
31	037	1F	00011111	US	Unit Separator

ASCII printable characters (character code 32-127)

Kode 32-127 biasa digunakan untuk semua variasi yang berbeda dari tabel ASCII, mereka disebut karakter yang dapat dicetak, mewakili huruf, angka, tanda baca, dan beberapa simbol lain-lain. Anda akan menemukan hampir setiap karakter di keyboard Anda. 127 Karakter merupakan perintah DEL.

DEC	OCT	HEX	BIN	Symbol	Description
32	040	20	00100000		Space
33	041	21	00100001	!	Exclamation mark
34	042	22	00100010	“	Double quotes (or speech marks)
35	043	23	00100011	#	Number
36	044	24	00100100	$	Dollar
37	045	25	00100101	%	Procenttecken
38	046	26	00100110	&	Ampersand
39	047	27	00100111	‘	Single quote
40	050	28	00101000	(	Open parenthesis (or open bracket)
41	051	29	00101001	)	Close parenthesis (or close bracket)
42	052	2A	00101010	*	Asterisk
43	053	2B	00101011	+	Plus
44	054	2C	00101100	,	Comma
45	055	2D	00101101	–	Hyphen
46	056	2E	00101110	.	Period, dot or full stop
47	057	2F	00101111	/	Slash or divide
48	060	30	00110000	0	Zero
49	061	31	00110001	1	One
50	062	32	00110010	2	Two
51	063	33	00110011	3	Three
52	064	34	00110100	4	Four
53	065	35	00110101	5	Five
54	066	36	00110110	6	Six
55	067	37	00110111	7	Seven

56	070	38	00111000	8	Eight
57	071	39	00111001	9	Nine
58	072	3A	00111010	:	Colon
59	073	3B	00111011	;	Semicolon
60	074	3C	00111100	<	Less than (or open angled bracket)
61	075	3D	00111101	=	Equals
62	076	3E	00111110	>	Greater than (or close angled bracket)
63	077	3F	00111111	?	Question mark
64	100	40	01000000	@	At symbol
65	101	41	01000001	A	Uppercase A
66	102	42	01000010	B	Uppercase B
67	103	43	01000011	C	Uppercase C
68	104	44	01000100	D	Uppercase D
69	105	45	01000101	E	Uppercase E
70	106	46	01000110	F	Uppercase F
71	107	47	01000111	G	Uppercase G
72	110	48	01001000	H	Uppercase H
73	111	49	01001001	I	Uppercase I
74	112	4A	01001010	J	Uppercase J
75	113	4B	01001011	K	Uppercase K
76	114	4C	01001100	L	Uppercase L
77	115	4D	01001101	M	Uppercase M
78	116	4E	01001110	N	Uppercase N
79	117	4F	01001111	O	Uppercase O
80	120	50	01010000	P	Uppercase P
81	121	51	01010001	Q	Uppercase Q
82	122	52	01010010	R	Uppercase R
83	123	53	01010011	S	Uppercase S
84	124	54	01010100	T	Uppercase T
85	125	55	01010101	U	Uppercase U
86	126	56	01010110	V	Uppercase V
87	127	57	01010111	W	Uppercase W
88	130	58	01011000	X	Uppercase X
89	131	59	01011001	Y	Uppercase Y
90	132	5A	01011010	Z	Uppercase Z
91	133	5B	01011011	[	Opening bracket
92	134	5C	01011100	\	Backslash
93	135	5D	01011101	]	Closing bracket
94	136	5E	01011110	^	Caret – circumflex
95	137	5F	01011111	_	Underscore
96	140	60	01100000	`	Grave accent
97	141	61	01100001	a	Lowercase a
98	142	62	01100010	b	Lowercase b
99	143	63	01100011	c	Lowercase c
100	144	64	01100100	d	Lowercase d
101	145	65	01100101	e	Lowercase e
102	146	66	01100110	f	Lowercase f
103	147	67	01100111	g	Lowercase g
104	150	68	01101000	h	Lowercase h
105	151	69	01101001	i	Lowercase i
106	152	6A	01101010	j	Lowercase j
107	153	6B	01101011	k	Lowercase k
108	154	6C	01101100	l	Lowercase l
109	155	6D	01101101	m	Lowercase m

110	156	6E	01101110	n	Lowercase n
111	157	6F	01101111	o	Lowercase o
112	160	70	01110000	p	Lowercase p
113	161	71	01110001	q	Lowercase q
114	162	72	01110010	r	Lowercase r
115	163	73	01110011	s	Lowercase s
116	164	74	01110100	t	Lowercase t
117	165	75	01110101	u	Lowercase u
118	166	76	01110110	v	Lowercase v
119	167	77	01110111	w	Lowercase w
120	170	78	01111000	x	Lowercase x
121	171	79	01111001	y	Lowercase y
122	172	7A	01111010	z	Lowercase z
123	173	7B	01111011	{	Opening brace
124	174	7C	01111100 |		Vertical bar
125	175	7D	01111101	}	Closing brace
126	176	7E	01111110	~	Equivalency sign – tilde
127	177	7F	01111111		Delete

The extended ASCII codes (character code 128-255)

Ada beberapa variasi tabel ASCII 8-bit. Tabel di bawah ini sesuai dengan ISO 8859-1, juga disebut ISO Latin-1. Kode 129-159 mengandung Microsoft ® Windows Latin-1 karakter diperpanjang.

DEC	OCT	HEX	BIN	Symbol	Description
128	200	80	10000000	€	Euro sign
129	201	81	10000001		
130	202	82	10000010	‚	Single low-9 quotation mark
131	203	83	10000011	ƒ	Latin small letter f with hook
132	204	84	10000100	„	Double low-9 quotation mark
133	205	85	10000101	…	Horizontal ellipsis
134	206	86	10000110	†	Dagger
135	207	87	10000111	‡	Double dagger
136	210	88	10001000	ˆ	Modifier letter circumflex accent
137	211	89	10001001	‰	Per mille sign
138	212	8A	10001010	Š	Latin capital letter S with caron
139	213	8B	10001011	‹	Single left-pointing angle quotation
140	214	8C	10001100	Œ	Latin capital ligature OE
141	215	8D	10001101		
142	216	8E	10001110	Ž	Latin captial letter Z with caron
143	217	8F	10001111		
144	220	90	10010000		
145	221	91	10010001	‘	Left single quotation mark
146	222	92	10010010	’	Right single quotation mark
147	223	93	10010011	“	Left double quotation mark
148	224	94	10010100	”	Right double quotation mark
149	225	95	10010101	•	Bullet
150	226	96	10010110	–	En dash
151	227	97	10010111	—	Em dash
152	230	98	10011000	˜	Small tilde

153	231	99	10011001	™	Trade mark sign
154	232	9A	10011010	š	Latin small letter S with caron
155	233	9B	10011011	›	Single right-pointing angle quotation mark
156	234	9C	10011100	œ	Latin small ligature oe
157	235	9D	10011101		
158	236	9E	10011110	ž	Latin small letter z with caron
159	237	9F	10011111	Ÿ	Latin capital letter Y with diaeresis
160	240	A0	10100000		Non-breaking space
161	241	A1	10100001	¡	Inverted exclamation mark
162	242	A2	10100010	¢	Cent sign
163	243	A3	10100011	£	Pound sign
164	244	A4	10100100	¤	Currency sign
165	245	A5	10100101	¥	Yen sign
166	246	A6	10100110	¦	Pipe, Broken vertical bar
167	247	A7	10100111	§	Section sign
168	250	A8	10101000	¨	Spacing diaeresis – umlaut
169	251	A9	10101001	©	Copyright sign
170	252	AA	10101010	ª	Feminine ordinal indicator
171	253	AB	10101011	«	Left double angle quotes
172	254	AC	10101100	¬	Not sign
173	255	AD	10101101		Soft hyphen
174	256	AE	10101110	®	Registered trade mark sign
175	257	AF	10101111	¯	Spacing macron – overline
176	260	B0	10110000	°	Degree sign
177	261	B1	10110001	±	Plus-or-minus sign
178	262	B2	10110010	²	Superscript two – squared
179	263	B3	10110011	³	Superscript three – cubed
180	264	B4	10110100	´	Acute accent – spacing acute
181	265	B5	10110101	µ	Micro sign
182	266	B6	10110110	¶	Pilcrow sign – paragraph sign
183	267	B7	10110111	·	Middle dot – Georgian comma
184	270	B8	10111000	¸	Spacing cedilla
185	271	B9	10111001	¹	Superscript one
186	272	BA	10111010	º	Masculine ordinal indicator
187	273	BB	10111011	»	Right double angle quotes
188	274	BC	10111100	¼	Fraction one quarter
189	275	BD	10111101	½	Fraction one half
190	276	BE	10111110	¾	Fraction three quarters
191	277	BF	10111111	¿	Inverted question mark
192	300	C0	11000000	À	Latin capital letter A with grave
193	301	C1	11000001	Á	Latin capital letter A with acute
194	302	C2	11000010	Â	Latin capital letter A with circumflex
195	303	C3	11000011	Ã	Latin capital letter A with tilde
196	304	C4	11000100	Ä	Latin capital letter A with diaeresis
197	305	C5	11000101	Å	Latin capital letter A with ring above
198	306	C6	11000110	Æ	Latin capital letter AE
199	307	C7	11000111	Ç	Latin capital letter C with cedilla
200	310	C8	11001000	È	Latin capital letter E with grave
201	311	C9	11001001	É	Latin capital letter E with acute
202	312	CA	11001010	Ê	Latin capital letter E with circumflex
203	313	CB	11001011	Ë	Latin capital letter E with diaeresis
204	314	CC	11001100	Ì	Latin capital letter I with grave
205	315	CD	11001101	Í	Latin capital letter I with acute
206	316	CE	11001110	Î	Latin capital letter I with circumflex

207	317	CF	11001111	Ï	Latin capital letter I with diaeresis
208	320	D0	11010000	Ð	Latin capital letter ETH
209	321	D1	11010001	Ñ	Latin capital letter N with tilde
210	322	D2	11010010	Ò	Latin capital letter O with grave
211	323	D3	11010011	Ó	Latin capital letter O with acute
212	324	D4	11010100	Ô	Latin capital letter O with circumflex
213	325	D5	11010101	Õ	Latin capital letter O with tilde
214	326	D6	11010110	Ö	Latin capital letter O with diaeresis
215	327	D7	11010111	×	Multiplication sign
216	330	D8	11011000	Ø	Latin capital letter O with slash
217	331	D9	11011001	Ù	Latin capital letter U with grave
218	332	DA	11011010	Ú	Latin capital letter U with acute
219	333	DB	11011011	Û	Latin capital letter U with circumflex
220	334	DC	11011100	Ü	Latin capital letter U with diaeresis
221	335	DD	11011101	Ý	Latin capital letter Y with acute
222	336	DE	11011110	Þ	Latin capital letter THORN
223	337	DF	11011111	ß	Latin small letter sharp s – ess-zed
224	340	E0	11100000	à	Latin small letter a with grave
225	341	E1	11100001	á	Latin small letter a with acute
226	342	E2	11100010	â	Latin small letter a with circumflex
227	343	E3	11100011	ã	Latin small letter a with tilde
228	344	E4	11100100	ä	Latin small letter a with diaeresis
229	345	E5	11100101	å	Latin small letter a with ring above
230	346	E6	11100110	æ	Latin small letter ae
231	347	E7	11100111	ç	Latin small letter c with cedilla
232	350	E8	11101000	è	Latin small letter e with grave
233	351	E9	11101001	é	Latin small letter e with acute
234	352	EA	11101010	ê	Latin small letter e with circumflex
235	353	EB	11101011	ë	Latin small letter e with diaeresis
236	354	EC	11101100	ì	Latin small letter i with grave
237	355	ED	11101101	í	Latin small letter i with acute
238	356	EE	11101110	î	Latin small letter i with circumflex
239	357	EF	11101111	ï	Latin small letter i with diaeresis
240	360	F0	11110000	ð	Latin small letter eth
241	361	F1	11110001	ñ	Latin small letter n with tilde
242	362	F2	11110010	ò	Latin small letter o with grave
243	363	F3	11110011	ó	Latin small letter o with acute
244	364	F4	11110100	ô	Latin small letter o with circumflex
245	365	F5	11110101	õ	Latin small letter o with tilde
246	366	F6	11110110	ö	Latin small letter o with diaeresis
247	367	F7	11110111	÷	Division sign
248	370	F8	11111000	ø	Latin small letter o with slash
249	371	F9	11111001	ù	Latin small letter u with grave
250	372	FA	11111010	ú	Latin small letter u with acute
251	373	FB	11111011	û	Latin small letter u with circumflex
252	374	FC	11111100	ü	Latin small letter u with diaeresis
253	375	FD	11111101	ý	Latin small letter y with acute
254	376	FE	11111110	þ	Latin small letter thorn
255	377	FF	11111111	ÿ	Latin small letter y with diaeresis

Assembly

No	Perintah	Keterangan
1.	ACALL (Absolute Call)	berfungsi untuk memanggil sub rutin program
2.	ADD (Add Immediate Data)	berfungsi untuk menambah 8 bit data langsung ke dalam isi akumulator dan menyimpan hasilnya pada akumulator
3.	ADDC (Add Carry Plus Immediate Data to Accumulator)	berfungsi untuk menambahkan isi carry flag (0 atau 1) ke dalam isi akumulator. Data langsung 8 bit ditambahkan ke akumulator.
4.	AJMP (Absolute Jump)	perintah jump mutlak. Jump dalam 2 KB dimulai dari alamat yang mengikuti perintah AJMP. AJMP berfungsi untuk mentransfer kendali program ke lokasi dimana alamat dikalkulasi dengan cara yang sama dengan perintah ACALL. Konter program ditambahkan dua kali dimana perintah AJMP adalah perintah 2-byte. Konter program di-load dengan a10 – a0 11 bits, untuk membentuk alamat tujuan 16-bit.
5.	ANL (logical AND memori ke akumulator)	berfunsi untuk mengAND-kan isi alamat data dengan isi akumulator.
6.	CJNE (Compare Indirect Address to Immediate Data)	berfungsi untuk membandingkan data langsung dengan lokasi memori yang dialamati oleh register R atau Akumulator A. apabila tidak sama maka instruksi akan menuju ke alamat kode. Format : CJNE R,#data,Alamat kode.
7.	CLR (Clear Accumulator)	berfungsi untuk mereset data akumulator menjadi 00H.
8.	CPL (Complement Accumulator)	berfungsi untuk mengkomplemen isi akumulator.
9.	DA (Decimal Adjust Accumulator)	berfungsi untuk mengatur isi akumulator ke padanan BCD, steleah penambahan dua angka BCD.
10.	DEC (Decrement Indirect Address)	berfungsi untuk mengurangi isi lokasi memori yang ditujukan oleh register R dengan 1, dan hasilnya disimpan pada lokasi tersebut
11.	DIV (Divide Accumulator by B)	berfungsi untuk membagi isi akumulator dengan isi register B. Akumulator berisi hasil bagi, register B berisi sisa pembagian.
12.	DJNZ (Decrement Register And Jump Id Not Zero)	DJNZ berfungsi untuk mengurangi nilai register dengan 1 dan jika hasilnya sudah 0 maka instruksi selanjutnya akan dieksekusi. Jika belum 0 akan menuju ke alamat kode.
13.	INC (Increment Indirect Address)	berfungsi untuk menambahkan isi memori dengan 1 dan menyimpannya pada alamat tersebut.
14.	JB (Jump if Bit is Set)	berfungsi untuk membaca data per satu bit, jika data tersebut adalah 1 maka akan menuju ke alamat kode dan jika 0 tidak akan menuju ke alamat kode.
15.	JBC (Jump if Bit Set and Clear Bit)	berfungsi sebagai perintah rel menguji yang terspesifikasikan secara bit. Jika bit di-set, maka Jump dilakukan ke alamat relatif dan yang terspesifikasi secara bit di dalam perintah dibersihkan. Segmen program berikut menguji bit yang kurang signifikan (LSB: Least Significant Byte), dan jika diketemukan bahwa ia telah di-set, program melompat ke READ lokasi. JBC juga berfungsi membersihkan LSB dari akumulator.

16.	JC (Jump if Carry is Set)	berfungsi untuk menguji isi carry flag. Jika berisi 1, eksekusi menuju ke alamat kode, jika berisi 0, instruksi selanjutnya yang akan dieksekusi.
17.	JMP (Jump to sum of Accumulator and Data Pointer)	berfungsi untuk memerintahkan loncat kesuato alamat kode tertentu. Format : JMP alamat kode
18.	JNB (Jump if Bit is Not Set)	berfungsi untuk membaca data per satu bit, jika data tersebut adalah 0 maka akan menuju ke alamat kode dan jika 1 tidak akan menuju ke alamat kode. Format : JNB alamat bit,alamat kode.
19.	JNC (Jump if Carry Not Set)	berfungsi untuk menguji bit Carry, dan jika tidak di-set, maka sebuah lompatan akan dilakukan ke alamat relatif yang telah ditentukan.
20.	JNZ (Jump if Accumulator Not Zero)	adalah mnemonik untuk instruksi jump if not zero (lompat jika tidak nol). Dalam hal ini suatu lompatan akan terjadi bilamana bendera nol dalam keadaan "clear", dan tidak akan terjadi lompatan bilamana bendera nol tersebut dalam keadaan set. Andaikan bahwa JNZ 7800H disimpan pada lokasi 2100H. Jika Z=0, instruksi berikutnya akan berasal dari lokasi 7800H: dan bilamana Z=1, program akan turun ke instruksi urutan berikutnya pada lokasi 2101H.
21.	JZ (Jump if Accumulator is Zero)	JZ berfungsi untuk menguji konten-konten akumulator. Jika bukan nol, maka lompatan dilakukan ke alamat relatif yang ditentukan dalam perintah.
22.	LCALL (Long Call)	berfungsi untuk memungkinkan panggilan ke subrutin yang berlokasi dimanapun dalam memori program 64K. Operasi LCALL berjalan seperti berikut: 1. Menambahkan ke dalam konter program sebanyak 3, karena perintahnya adalah perintah 3-byte. 2. Menambahkan penunjuk stack sebanyak 1. 3. Menyimpan byte yang lebih rendah dari konter program ke dalam stack. 4. Menambahkan penunjuk stack. 5. Menyimpan byte yang lebih tinggi dari program ke dalam stack. 6. Me-load konter program dengan alamat tujuan 16-bit.
23.	LJMP (Long Jump)	Long Jump befungsi untuk memungkinkan lompatan tak bersyarat kemana saja dalam lingkup ruang memori program 64K. LCALL adalah perintah 3-byte. Alamat tujuan 16-bit ditentukan secara langsung dalam perintah tersebut. Alamat tujuan ini di-load ke dalam konter program oleh perintah LJMP.
24.	MOV (Move From Memory)	berfungsi untuk memindahkan isi akumulator/register atau data dari nilai luar atau alamat lain.
25.	MOVC (Move From Codec Memory)	berfungsi untuk mengisi accumulator dengan byte kode atau konstanta dari program memory. Alamat byte tersebut adalah hasil penjumlahan unsigned 8 bit pada accumulator dan 16 bit register basis yang dapat berupa data pointer atau program counter. Instruksi ini tidak mempengaruhi flag apapun juga.

26.	MOVX (Move Accumulator to External Memory Addressed by Data Pointer)	berfungsi untuk memindahkan isi akumulator ke memori data eksternal yang alamatnya ditunjukkan oleh isi data pointer.
27.	MUL (Multiply)	MUL AB berfungsi untuk mengalikan unsigned 8 bit integer pada accumulator dan register B. Byte rendah (low order) dari hasil perkalian akan disimpan dalam accumulator sedangkan byte tinggi (high order) akan disimpan dalam register B. Jika hasil perkalian lebih besar dari 255 (0FFh), overflow flag akan bernilai '1'. Jika hasil perkalian lebih kecil atau sama dengan 255, overflow flag akan bernilai '0'. Carry flag akan selalu dikosongkan.
28.	NOP (No Operation)	Fungsi NOP adalah eksekusi program akan dilanjutkan ke instruksi berikutnya. Selain PC, instruksi ini tidak mempengaruhi register atau flag apapun juga.
29.	ORL (Logical OR Immediate Data to Accumulator)	berfungsi sebagai instruksi Gerbang logika OR yang akan menjumlahkan Accumulator terhadap nilai yang ditentukan. Format : ORL A,#data.
30.	POP (Pop Stack to Memory)	Instruksi POP berfungsi untuk menempatkan byte yang ditunjukkan oleh stack pointer ke suatu alamat data.
31.	PUSH (Push Memory onto Stack)	berfungsi untuk menaikkan stack pointer kemudian menyimpan isinya ke suatu alamat data pada lokasi yang ditunuk oleh stack pointer.
32.	RET (Return from subroutine)	berfungsi untuk kembali dari suatu subrutin program ke alamat terakhir subrutin tersebut di panggil.
33.	RETI (Return From Interrupt)	berfungsi untuk mengambil nilai byte tinggi dan rendah dari PC dari stack dan mengembalikan kondisi logika interrupt agar dapat menerima interrupt lain dengan prioritas yang sama dengan prioritas interrupt yang baru saja diproses. Stack pointer akan dikurangi dengan 2. Instruksi ini tidak mempengaruhi flag apapun juga. Nilai PSW tidak akan dikembalikan secara otomatis ke kondisi sebelum interrupt. Eksekusi program akan dilanjutkan pada alamat yang diambil tersebut. Umumnya alamat tersebut adalah alamat setelah lokasi dimana terjadi interrupt. Jika interrupt dengan prioritas sama atau lebih rendah tertunda saat RETI dieksekusi, maka satu instruksi lagi akan dieksekusi sebelum interrupt yang tertunda tersebut diproses.
34.	RL (Rotate Accumulator Left)	berfungsi untuk memutar setiap bit dalam akumulator satu posisi ke kiri.
35.	RLC (Rotate Left through Carry)	Memutar (Rotate) Accumulator ke Kiri (Left) Melalui Carry Flag. Kedelapan bit accumulator dan carry flag akan diputar satu bit ke kiri secara bersama-sama. Bit 7 akan dirotasi ke carry flag, nilai carry flag akan berpindah ke posisi bit 0. Instruksi ini tidak mempengaruhi flag lain.
36.	RR (Rotate Right)	Memutar (Rotate) Accumulator ke Kanan (Right). Kedelapan bit accumulator akan diputar satu bit ke kanan. Bit 0 akan dirotasi ke posisi bit 7. Instruksi ini tidak mempengaruhi flag apapun juga.
37.	RRC (Rotate Right through Carry)	Memutar (Rotate) Accumulator ke Kanan (Right) Melalui Carry Flag. Kedelapan bit accumulator dan carry flag akan diputar satu bit ke kanan secara

		bersama-sama. Bit 0 akan dirotasi ke carry flag, nilai carry flag akan berpindah ke posisi bit 7. Instruksi ini tidak mempengaruhi flag lain.
38.	SETB (set Carry flag)	berfungsi untuk menset carry flag.
39.	SJMP (Short Jump)	berfungsi untuk mentransfer kendali ke alamat tujuan dalam 127 bytes yang mengikuti dan 128 yang mengawali perintah SJMP. Alamat tujuannya ditentukan sebagai sebuat alamat relative 8-bit. Ini adalah Jump tidak bersyarat. Perintah SJMP menambahkan konter program sebanyak 2 dan menambahkan alamat relatif ke dalamnya untuk mendapatkan alamat tujuan. Alamat relatif tersebut ditentukan dalam perintah sebagai 'SJMP rel'.
40.	SUBB (Subtract With Borrow)	Pengurangan (Subtract) dengan Peminjaman (Borrow). SUBB mengurangi variabel yang tertera pada operand kedua dan carry flag sekaligus dari accumulator dan menyimpan hasilnya pada accumulator. SUBB akan memberi nilai '1' pada carry flag jika peminjaman ke bit 7 dibutuhkan dan mengosongkan C jika tidak dibutuhkan peminjaman. Jika C bernilai '1' sebelum mengeksekusi SUBB, hal ini menandakan bahwa terjadi peminjaman pada proses pengurangan sebelumnya, sehingga carry flag dan source byte akan dikurangkan dari accumulator secara bersama-sama. AC akan bernilai '1' jika peminjaman ke bit 3 dibutuhkan dan mengosongkan AC jika tidak dibutuhkan peminjaman. OV akan bernilai '1' jika ada peminjaman ke bit 6 namun tidak ke bit 7 atau ada peminjaman ke bit 7 namun tidak ke bit 6. Saat mengurangi signed integer, OV menandakan adanya angka negative sebagai hasil dari pengurangan angka negatif dari angka positif atau adanya angka positif sebagai hasil dari pengurangan angka positif dari angka negative. Addressing mode yang dapat digunakan adalah: register, direct, register indirect, atau immediate data.
41.	SWAP (Swap Nibbles)	Menukar (Swap) Upper Nibble dan Lower Nibble dalam Accumulator. SWAP A akan menukar nibble (4 bit) tinggi dan nibble rendah dalam accumulator. Operasi ini dapat dianggap sebagai rotasi 4 bit dengan RR atau RL. Instruksi ini tidak mempengaruhi flag apapun juga.
42.	XCH (Exchange Bytes)	Menukar (Exchange) Accumulator dengan Variabel Byte. XCH akan mengisi accumulator dengan variabel yang tertera pada operand kedua dan pada saat yang sama juga akan mengisikan nilai accumulator ke dalam variabel tersebut. Addressing mode yang dapat digunakan adalah: register, direct, atau register indirect.
43.	XCHD (Exchange Digits)	Menukar (Exchange) Digit. XCHD menukar nibble rendah dari accumulator, yang umumnya mewakili angka heksadesimal atau BCD, dengan nibble rendah dari internal data memory yang diakses secara indirect. Nibble tinggi kedua register tidak akan terpengaruh. Instruksi ini tidak mempengaruhi flag apapun juga.

44.	XRL (Exclusive OR Logic)	Logika Exclusive OR untuk Variabel Byte XRL akan melakukan operasi bitwise logika exclusive OR antara kedua variabel yang dinyatakan. Hasilnya akan disimpan pada destination byte. Instruksi ini tidak mempengaruhi flag apapun juga. Kedua operand mampu menggunakan enam kombinasi addressing mode. Saat destination byte adalah accumulator, source byte dapat berupa register, direct, register indirect, atau immediate data. Saat destination byte berupa direct address, source byte dapat berupa accumulator atau immediate data.